Heriot-Watt University
Research Gateway

An Efficient v-minimum Absolute Deviation Distribution Regression Machine

Citation for published version:
Wang, Y, Wang, Y, Song, Y, Xie, X, Huang, L, Pang, W & Coghill, GM 2020, 'An Efficient v-minimum Absolute Deviation Distribution Regression Machine', *IEEE Access*, vol. 8, pp. 85533-85551. https://doi.org/10.1109/ACCESS.2020.2992703

Digital Object Identifier (DOI):
10.1109/ACCESS.2020.2992703

Link:
Link to publication record in Heriot-Watt Research Portal

Document Version:
Publisher's PDF, also known as Version of record

Published In:
IEEE Access

General rights
Copyright for the publications made accessible via Heriot-Watt Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
Heriot-Watt University has made every reasonable effort to ensure that the content in Heriot-Watt Research Portal complies with UK legislation. If you believe that the public display of this file breaches copyright please contact open.access@hw.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 26, 2020, accepted April 30, 2020, date of publication May 6, 2020, date of current version May 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992703

An Efficient v-Minimum Absolute Deviation Distribution Regression Machine

YAN WANG[1], YAO WANG[1], YINGYING SONG[1], XUPING XIE[1], LAN HUANG[1,2], WEI PANG[3,4], AND GEORGE M. COGHILL[4]

[1]Key Laboratory of Symbol Computation and Knowledge Engineering, Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Zhuhai Laboratory of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Department of Computer Science and Technology, Zhuhai College, Jilin University, Zhuhai 519041, China
[3]School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, U.K.
[4]Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, U.K.

Corresponding authors: Lan Huang (huanglan@jlu.edu.cn) and Wei Pang (w.pang@hw.ac.uk)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702214 and Grant 61772227, in part by the Development Project of Jilin Province of China under Grant 20180414012GH, Grant 20180201045GX, and Grant 2020C003, in part by the Guangdong Key Project for Applied Fundamental Research under Grant 2018KZDXM076, and in part by the Jilin Provincial Key Laboratory of Big Date Intelligent Computing under Grant 20180622002JC.

ABSTRACT Support Vector Regression (SVR) and its variants are widely used regression algorithms, and they have demonstrated high generalization ability. This research proposes a new SVR-based regressor : v-minimum absolute deviation distribution regression (v-MADR) machine. Instead of merely minimizing structural risk, as with v-SVR, v-MADR aims to achieve better generalization performance by minimizing both the absolute regression deviation mean and the absolute regression deviation variance, which takes into account the positive and negative values of the regression deviation of sample points. For optimization, we propose a dual coordinate descent (DCD) algorithm for small sample problems, and we also propose an averaged stochastic gradient descent (ASGD) algorithm for large-scale problems. Furthermore, we study the statistical property of v-MADR that leads to a bound on the expectation of error. The experimental results on both artificial and real datasets indicate that our v-MADR has significant improvement in generalization performance with less training time compared to the widely used v-SVR, LS-SVR, ε-TSVR, and linear ε-SVR. Finally, we open source the code of v-MADR at https://github.com/AsunaYY/v-MADR for wider dissemination.

INDEX TERMS v-support vector regression, absolute regression deviation mean, absolute regression deviation variance, dual coordinate descent algorithm.

I. INTRODUCTION

Support vector regression (SVR) [1]–[3] has been widely used in machine learning, since it can achieve better structural risk minimization. SVR realizes linear regression mainly by constructing linear decision functions in high dimensional space. Compared with other regression methods, such as least square regression [4], Neural Networks (NN) regression [5], logistic regression [6], and ridge regression [7], SVR has better generalization ability for regression problems [8]–[10]. In recent years, there have been many studies about SVR-based algorithms. Several SVR approaches have been developed, such as ε-support vector regression (ε-SVR) [1], [11], v-support vector regression (v-SVR) [12], and least square support vector regression (LS-SVR) [13], [14]. The basic idea of these methods is to find the decision function by maximizing the boundaries of two parallel hyperplanes. Different from ε-SVR, v-SVR introduces another parameter, v, to control the number of support vectors and adjust the parameter ε automatically. The parameter v has a certain range of values, that is, (0,1]. When solving the quadratic specification problem (QPP), v-SVR reduces the number of computational parameters by half, which greatly reduces the computational complexity. Besides, some researchers have proposed the non-parallel planar regressors, such as twin support vector regression (TSVR) [15], ε-twin support vector regression (ε-TSVR) [16], parametric-insensitive nonparallel support vector regression (PINSVR) [17], lagrangian

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamad Forouzanfar.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

support vector regression [18], and lagrangian twin support vector regression (LTSVR) [19]. These algorithms demonstrate good ability to capture data structure and boundary information.

Support vector (SV) theory indicates that maximizing the minimum margin is not the only way to construct the separating hyperplane for SVM. Zhang and Zhou [20], Zhou [21], Zhang and Zhou [22], and Gao and Zhou [23] proposed the large margin distribution machine (LDM), which was designed to maximize the margin mean and minimize the margin variance simultaneously. Gao and Zhou [23] proved that the margin distribution including the margin mean and the margin variance was crucial for generalization compared to a single margin, and optimizing the margin distribution can also naturally accommodate class imbalance and unequal misclassification costs [21]. Inspired by the idea of LDM, Liu *et al.* proposed a minimum deviation distribution regression (MDR) [24], which introduced the statistics of regression deviation into ε-SVR. More specifically, MDR minimizes the regression deviation mean and the regression deviation variance while optimizing the minimum margin. In addition, Reshma and Pritam were also inspired by the idea of LDM, and they proposed a large-margin distribution machine-based regression model (LDMR) and a new loss function [25], [26]. However, the definition of the deviation mean in MDR is not very appropriate for positive and negative samples, and the speed of ε-SVR strategy that MDR used can be further improved.

Considering the above advances in SVR, in this research, we introduce the statistical information into v-SVR and propose an v-minimum absolute deviation distribution regression (v-MADR). We give the definition of regression deviation mean which takes into account both the positive and negative values of the regression deviation of sample points. Inspired by recent theoretical results [20]–[24], v-MADR simultaneously minimizes the absolute regression deviation mean and the absolute regression deviation variance based on the v-SVR strategy, thereby greatly improving the generalization performance [21], [23]. To solve the optimization problem, we propose a dual coordinate descent (DCD) algorithm for small sample problems, and we also propose an averaged stochastic gradient descent (ASGD) algorithm for large-scale problems. Furthermore, the boundary on error expectation of v-MADR is studied. The performance of v-MADR is assessed on both artificial and real datasets in comparison with other typical regression algorithms, such as v-SVR, LS-SVR, ε-TSVR, and linear ε-SVR. According to previous research, SVR-based algorithms show better generalization ability for regression problems [8]–[10]. In conclusion, our experimental results demonstrate that the proposed v-MADR can lead to better performance than other algorithms for regression problems. The main contributions of this paper are as follows:

1) We propose a new regression algorithm that minimizes both the absolute regression deviation mean and the absolute regression deviation variance, and this new algorithm takes into account the positive and negative values of the regression deviation of sample points.
2) We propose two optimization algorithms, i.e., the dual coordinate descent (DCD) algorithm for small samples problems and the averaged stochastic gradient descent (ASGD) algorithm for large-scale problems.
3) We theoretically prove the upper bound on the generalization error of v-MADR and analyze the computational complexity of our optimization algorithms.

As SVR-based algorithms are widely used for regression problems, v-MADR has great application potential.

The rest of this paper is organized as follows: Section 2 introduces the notations used in this paper and presents a brief review of SVR as well as the recent progress in SV theory. Section 3 introduces the proposed v-MADR, including the kernel and the bound on the expectation of error. Experimental results are reported in Section 4, and finally, the conclusions are drawn in Section 5.

II. BACKGROUND

Suppose $\boldsymbol{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ is a training set of n samples, where $\boldsymbol{x}_i \in \boldsymbol{\chi}$ is the input sample in the form of d-dimensional vectors and $y_i \in R$ is the corresponding target value. The objective function is $f(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b$, where $\boldsymbol{w} \in R^m$ is the weight vector, $b \in R$ is the bias term, and $\phi(\boldsymbol{x})$ is the mapping function induced by a kernel κ, i.e., $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$. To reduce the complexity brought by b, we enlarge the dimension of $\boldsymbol{w}$ and $\phi(\boldsymbol{x}_i)$ as in [27], i.e., $\boldsymbol{w} \leftarrow [\boldsymbol{w}, b]^T$, $\phi(\boldsymbol{x}_i) \leftarrow [\phi(\boldsymbol{x}_i), 1]$. Thus, the function $f(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b$ becomes the following form:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}).$$

In what follows, we only consider problems in the form of the above function.

Formally, we denote $\boldsymbol{X}$ as the matrix whose i-th column is $\phi(\boldsymbol{x}_i)$, i.e., $\boldsymbol{X} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]$, and $\boldsymbol{y} = [y_1, \ldots, y_n]^T$ is a column vector.

A. THE SVR ALGORITHMS

There are two traditional methods for solving support vector regression (SVR) algorithms, namely ε-SVR [1], [11] and v-SVR [12]. In order to find the best fitting surface, ε-SVR maximizes the minimum margin containing the data in the so-called ε-tube, in which the distances of the data to the fitting hyperplane are not larger than ε. Therefore, ε-SVR with soft-margin can be expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\xi}^*} \ & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\left(\boldsymbol{e}^T\boldsymbol{\xi} + \boldsymbol{e}^T\boldsymbol{\xi}^*\right) \\ s.t. \ & \boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w} \le \varepsilon\boldsymbol{e} + \boldsymbol{\xi}, \\ & \boldsymbol{X}^T\boldsymbol{w} - \boldsymbol{y} \le \varepsilon\boldsymbol{e} + \boldsymbol{\xi}^*, \\ & \boldsymbol{\xi}, \boldsymbol{\xi}^* \ge \boldsymbol{0}, \end{aligned}$$

where parameter C is used for the tradeoff between the flatness of $f(\boldsymbol{x})$ and the tolerance of the deviation larger than ε; $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_n]$ and $\boldsymbol{\xi}^* = \left[\xi_1^*, \xi_2^*, \ldots, \xi_n^*\right]$ are the slack

variables measuring the distances of the training samples outside the ε-tube from the ε-tube itself as soft-margin; $\boldsymbol{e}$ stands for the all-one vector of appropriate dimensions.

The dual problem of ε-SVR is formulated as

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)^T \boldsymbol{Q}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)+\varepsilon\left(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*\right)+\boldsymbol{y}^T\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right) \\ &s.t.\ \boldsymbol{e}^T\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)=0, \quad 0\le\alpha_i,\alpha_i^*\le C,\ i=1,2,\ldots,n, \end{aligned} \tag{1}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ are the Lagrange multipliers;$Q_{ij} = \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$.

In order to facilitate the calculation, Formula (1) can be transformed as follows:

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}\tilde{\alpha}^T\begin{bmatrix}\boldsymbol{Q} & -\boldsymbol{Q}\\ -\boldsymbol{Q} & \boldsymbol{Q}\end{bmatrix}\tilde{\alpha}+\begin{bmatrix}\varepsilon\boldsymbol{e}+\boldsymbol{y}\\ \varepsilon\boldsymbol{e}-\boldsymbol{y}\end{bmatrix}^T\tilde{\alpha} \\ &s.t.\ \begin{bmatrix}\boldsymbol{e}\\ -\boldsymbol{e}\end{bmatrix}^T\tilde{\alpha}=0, \quad 0\le\tilde{\alpha}_i\le C,\ i=1,2,\ldots,2n, \end{aligned} \tag{2}$$

where $\tilde{\alpha}=[\boldsymbol{\alpha}^T,\boldsymbol{\alpha}^{*T}]^T$.

v-SVR [12] is another commonly used algorithm for solving SVR. Compared with ε-SVR, *v*-SVR uses a new parameter $v \in (0,1]$ to control the number of support vectors and training errors and adjust parameter ε automatically. According to Gu *et al.*, the objective function $f(\boldsymbol{x})$ in *v*-SVR is represented by the following constrained minimization problem with soft-margin [28]–[30]:

$$\begin{aligned} &\min_{\boldsymbol{w},\varepsilon,\boldsymbol{\xi},\boldsymbol{\xi}^*} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+C\left(v\varepsilon+\frac{1}{n}\left(\boldsymbol{e}^T\boldsymbol{\xi}+\boldsymbol{e}^T\boldsymbol{\xi}^*\right)\right) \\ &s.t.\ \boldsymbol{y}-\boldsymbol{X}^T\boldsymbol{w}\le\varepsilon\boldsymbol{e}+\boldsymbol{\xi}, \\ &\quad \boldsymbol{X}^T\boldsymbol{w}-\boldsymbol{y}\le\varepsilon\boldsymbol{e}+\boldsymbol{\xi}^*, \\ &\quad \boldsymbol{\xi},\boldsymbol{\xi}^*\ge\boldsymbol{0},\varepsilon\boldsymbol{e}\ge\boldsymbol{0}. \end{aligned}$$

The dual problem of *v*-SVR is

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)^T\boldsymbol{Q}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)+\boldsymbol{y}^T\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right) \\ &s.t.\ \boldsymbol{e}^T\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)=0, \boldsymbol{e}^T\left(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*\right)\le Cv, \\ &\quad 0\le\alpha_i,\alpha_i^*\le\frac{C}{n}, \quad i=1,2,\ldots,n. \end{aligned}$$

According to Chang *et al.* and Crisp *et al.*, the inequality $\boldsymbol{e}^{\mathrm{T}}(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)\le Cv$ in *v*-SVR can be replaced by the equality form of $\boldsymbol{e}^{\mathrm{T}}(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)=Cv$ with the constraint $0<v\le 1$ [11], [31], so we have

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)^T\boldsymbol{Q}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)+\boldsymbol{y}^T\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right) \\ &s.t.\ \boldsymbol{e}^T\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)=0, \quad \boldsymbol{e}^T\left(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*\right)=Cv, \\ &\quad 0\le\alpha_i,\alpha_i^*\le\frac{C}{n}, \quad i=1,2,\ldots,n. \end{aligned} \tag{3}$$

We substitute the equation $\boldsymbol{\alpha}^*=Cv\boldsymbol{e}-\boldsymbol{\alpha}$ into Formula (3), and Formula (3) can be written as follows:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \frac{1}{2}(2\boldsymbol{\alpha}-Cv\boldsymbol{e})^T\boldsymbol{Q}(2\boldsymbol{\alpha}-Cv\boldsymbol{e})+\boldsymbol{y}^T(2\boldsymbol{\alpha}-Cv\boldsymbol{e}) \\ &s.t.\ \boldsymbol{e}^T(2\boldsymbol{\alpha}-Cv\boldsymbol{e})=0, \\ &\quad 0\le\alpha_i\le\frac{C}{n}, \quad i=1,2,\ldots,n. \end{aligned} \tag{4}$$

As one can see from Formula (2) and (4), by substituting the equation $\boldsymbol{\alpha}^*=Cv\boldsymbol{e}-\boldsymbol{\alpha}$ into the dual problem, the number of computational parameters of the *v*-SVR has been reduced by half compared to ε-SVR when solving the QPP. The difference in both time complexity and spatial complexity between ε-SVR and *v*-SVR can be expressed as follows:

$$\frac{O(\varepsilon-SVR)}{O(v-SVR)}=O\left(\frac{Formula\,(2)}{Formula\,(4)}\right)=O\left(\frac{2n*2n}{n*n}\right)=4.$$

B. RECENT PROGRESS IN SV THEORY

Recent SV theory indicates that maximizing the minimum margin is not the only way to construct the separating hyperplane for SVR, because it does not necessarily lead to better generalization performance [20]. There may exist the so-called data piling problem in SVR [32], that is, the separating hyperplane produced by SVR tends to maximize data piling, which makes the data pile together when they are projected onto the hyperplane. If the distribution of the boundary data is different from that of the internal data, the hyperplane constructed by SVR will be inconsistent with the actual data distribution, which reduces the performance of SVR.

Fortunately, Gao and Zhou have demonstrated that marginal distribution was critical to the generalization performance [23]. By using the margin mean and the margin variance, the model is robust to different distributions of boundary data and noise. Inspired by the above research, MDR [24] introduced the statistics of deviation into ε-SVR and this allows more data to have impact on the construction of the hyperplane.

In MDR, the regression deviation of sample $(\boldsymbol{x}_i,y_i)$ is formulated as

$$\gamma_i=y_i-f(\boldsymbol{x}_i), \quad \forall i=1,\ldots,n. \tag{5}$$

So, the regression deviation mean is

$$\bar{\gamma}=\frac{1}{n}\sum_{i=1}^{n}\gamma_i=\frac{1}{n}\sum_{i=1}^{n}(y_i-f(\boldsymbol{x}_i))=\frac{1}{n}\boldsymbol{e}^T\left(\boldsymbol{y}-\boldsymbol{X}^T\boldsymbol{w}\right),$$

and the regression deviation variance is defined as

$$\begin{aligned} \hat{\gamma}^2&=\left(\frac{1}{n}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left[y_i-f(\boldsymbol{x}_i)-y_j+f(\boldsymbol{x}_j)\right]^2}\right)^2 \\ &=\frac{1}{n^2}\left\{2\left[\boldsymbol{w}^T\boldsymbol{X}\left(n\boldsymbol{I}-\boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{X}^T\boldsymbol{w}-2\boldsymbol{y}^T\left(n\boldsymbol{I}-\boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{X}^T\boldsymbol{w}\right.\right. \\ &\quad\left.\left.+\boldsymbol{y}^T\left(n\boldsymbol{I}-\boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{y}\right]\right\}. \end{aligned}$$

MDR minimizes the regression deviation mean and the regression deviation variance simultaneously, so we have the following primal problem of soft-margin MDR:

$$\min_{\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\xi}^*} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+\lambda_1\hat{\gamma}^2+\lambda_2\bar{\gamma}^2+C\left(\boldsymbol{e}^T\boldsymbol{\xi}+\boldsymbol{e}^T\boldsymbol{\xi}^*\right)$$

$$
\begin{aligned}
s.t.\ & \boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w} \le \varepsilon\boldsymbol{e} + \boldsymbol{\xi}, \\
& \boldsymbol{w} - \boldsymbol{y} \le \varepsilon\boldsymbol{e} + \boldsymbol{\xi}^*, \\
& \boldsymbol{\xi}, \boldsymbol{\xi}^* \ge \boldsymbol{0},
\end{aligned}
$$

where λ_1 and λ_2 are the parameters for trading-off the regression deviation variance, the regression deviation mean and the model complexity.

Here, we can see from Equation (5) that the regression deviation, γ_i, is positive when the sample $(\boldsymbol{x}_i, y_i)$ lies above the regressor and negative when the sample $(\boldsymbol{x}_i, y_i)$ lies under the regressor. But in fact, for regression, the regression deviation of the sample $(\boldsymbol{x}_i, y_i)$ is the distance between the actual value and the estimated one, that is, $\gamma_i = |y_i - f(\boldsymbol{x}_i)|, \forall i = 1, \ldots, n$. Therefore, the definition of the deviation mean in MDR here is not very appropriate.

On the other hand, when solving QPP, MDR uses the ε-SVR strategy, and it needs to calculate $2n$ (n is the number of training samples) parameters. Calculating a large number of parameters will increase the computational complexity and reduce the speed of the algorithm. Considering this, in the remainder of this paper, we will introduce our latest advances in SV theory and address the limitations of ε-SVR strategy.

III. ν-MININUM ABSOLUTE DEVIATION DISTRIBUTION REGRESSION

In this section, we first formulate the absolute deviation distribution which takes into account the positive and negative values of the regression deviation of samples. Then we give the optimization algorithms and the theoretical proof.

A. FORMULATION OF ν-MADR

The two most straightforward statistics for characterizing the absolute deviation distribution are the mean and the variance of absolute deviation. In regression problems, the absolute regression deviation of sample $(\boldsymbol{x}_i, y_i)$ is formulated as

$$
\varphi_i = |y_i - f(\boldsymbol{x}_i)|, \quad \forall i = 1, \ldots, n. \tag{6}
$$

φ_i is actually the distance between the actual value of the sample $(\boldsymbol{x}_i, y_i)$ and the estimated one. According to the definition in Equation (6), we give the definitions of statistics of absolute deviation in regression.

Definition 1: Absolute regression deviation mean is defined as follows:

$$
\begin{aligned}
\bar{\varphi} &= \frac{1}{n}\sum_{i=1}^{n}\varphi_i^2 = \frac{1}{n}\sum_{i=1}^{n}|y_i - f(\boldsymbol{x}_i)|^2 \\
&= \frac{1}{n}\left(\boldsymbol{w}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w} - 2\boldsymbol{y}^T\boldsymbol{X}^T\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{y}\right).
\end{aligned} \tag{7}
$$

The absolute regression deviation mean actually represents the expected value of difference between the actual values of data and the estimated ones. In order to facilitate the calculation, we have done a square process in this definition. In fact, we can view the absolute regression deviation mean as the adjusted distances of data to their fitting hyperplane. Next, we give the concept of the absolute regression deviation variance as follows:

Definition 2: Absolute regression deviation variance is defined as follows:

$$
\begin{aligned}
\hat{\varphi} &= \left(\frac{1}{n}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left|y_i - f(\boldsymbol{x}_i) - y_j + f(\boldsymbol{x}_j)\right|^2}\right)^2 \\
&= \frac{2}{n^2}\left[\boldsymbol{w}^T\boldsymbol{X}\left(n\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{X}^T\boldsymbol{w} - 2\boldsymbol{y}^T\left(n\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{X}^T\boldsymbol{w}\right. \\
&\quad \left. + \boldsymbol{y}^T\left(n\boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{y}\right].
\end{aligned} \tag{8}
$$

We can see that the absolute regression deviation variance quantifies the scatter of regression.

Existing SVR's loss is calculated only if the absolute value of the difference between the actual data and the estimated values is greater than a threshold. The fitting hyperplane constructed by SVR is only affected by the distribution of the boundary data. If the distribution of the boundary data largely deviates from that of the internal data, the hyperplane constructed will be inconsistent with the actual overall data distribution. To overcome this issue, ν-MADR aims to obtain a tradeoff between the distribution of the boundary data and that of the internal data. This means that the fitting hyperplane constructed by ν-MADR is not only determined by the distribution of the boundary data, but also measures the influence of the overall data distribution on the fitting hyperplane by simultaneously minimizing the absolute regression deviation mean and the absolute regression deviation variance, which is closer to the real distribution for many datasets and is more robust to noise.

Therefore, similar to the soft-margin of ν-SVR [28], the final optimization problem considering the soft-margin has the following form:

$$
\begin{aligned}
&\min_{\boldsymbol{w},\varepsilon,\boldsymbol{\xi},\boldsymbol{\xi}^*} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \lambda_1\hat{\varphi} + \lambda_2\bar{\varphi} + C\left(\nu\varepsilon + \frac{1}{n}\left(\boldsymbol{e}^T\boldsymbol{\xi} + \boldsymbol{e}^T\boldsymbol{\xi}^*\right)\right) \\
&s.t.\ \boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w} \le \varepsilon\boldsymbol{e} + \boldsymbol{\xi}, \\
&\quad \boldsymbol{X}^T\boldsymbol{w} - \boldsymbol{y} \le \varepsilon\boldsymbol{e} + \boldsymbol{\xi}^*, \\
&\quad \boldsymbol{\xi}, \boldsymbol{\xi}^* \ge \boldsymbol{0}, \quad \varepsilon\boldsymbol{e} \ge \boldsymbol{0},
\end{aligned} \tag{9}
$$

where parameters λ_1 and λ_2 are aimed at achieving the tradeoff among the absolute regression deviation mean, the absolute regression deviation variance and the model complexity. It is evident that the soft-margin ν-MADR subsumes the soft-margin ν-SVR when λ_1 and λ_2 both equal 0. The meanings of the other variables have been introduced in previous formula.

B. ALGORITHMS FOR ν-MADR

Solving Formula (9) is a key point for ν-MADR in practical use. In this section, we first design a dual coordinate descent (DCD) algorithm for kernel ν-MADR, and then present an average stochastic gradient descent (ASGD) algorithm for large-scale linear kernel ν-MADR.

1) KERNEL ν-MADR

By substituting the absolute regression deviation mean $\hat{\varphi}$ (Definition 1) and the absolute regression deviation variance

$\hat{\varphi}$ (Definition 2) into Formula (9), we obtain Formula (10) as follows:

$$\begin{aligned}\min_{\boldsymbol{w},\varepsilon,\boldsymbol{\xi},\boldsymbol{\xi}^*}\ & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+\boldsymbol{w}^T\boldsymbol{X}\left(\frac{2\lambda_1+\lambda_2}{n}\boldsymbol{I}-\frac{2\lambda_1}{n^2}\boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{X}^T\boldsymbol{w}\\ &-\left(\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{y}^T-\frac{4\lambda_1}{n^2}\boldsymbol{y}^T\boldsymbol{e}\boldsymbol{e}^T\right)\boldsymbol{X}^T\boldsymbol{w}\\ &+C\left(v\varepsilon+\frac{1}{n}\left(\boldsymbol{e}^T\boldsymbol{\xi}+\boldsymbol{e}^T\boldsymbol{\xi}^*\right)\right)\\ s.t.\ & \boldsymbol{y}-\boldsymbol{X}^T\boldsymbol{w}\le\varepsilon\boldsymbol{e}+\boldsymbol{\xi},\\ & \boldsymbol{X}^T\boldsymbol{w}-\boldsymbol{y}\le\varepsilon\boldsymbol{e}+\boldsymbol{\xi}^*,\\ & \boldsymbol{\xi},\boldsymbol{\xi}^*\ge\boldsymbol{0},\\ & \varepsilon\boldsymbol{e}\ge\boldsymbol{0},\end{aligned}\tag{10}$$

The $\boldsymbol{y}\boldsymbol{y}^T$ and $\boldsymbol{y}^T\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{y}$ terms in $\bar{\varphi}$ (Definition 1) and $\hat{\varphi}$ (Definition 2) are constants in an optimization problem, so we omit this term. However, Formula (10) is still intractable because of the high dimensionality of $\phi(\boldsymbol{x})$ and its complicated form. Inspired by [20], [33], we give the following theorem to state the optimal solution $\boldsymbol{w}$ for Formula (10).

Theorem 1: The optimal solution $\boldsymbol{w}$ for Formula (10) can be represented by the following form:

$$\boldsymbol{w}=\sum_{i=1}^{n}\left(\alpha_i-\alpha_i^*\right)\phi\left(\boldsymbol{x}_i\right)=\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right),\tag{11}$$

where $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\ldots,\alpha_n]^T$ and $\boldsymbol{\alpha}^*=[\alpha_1^*,\alpha_2^*,\ldots,\alpha_n^*]^T$ are the parameters of v-MADR.

Proof: Suppose that $\boldsymbol{w}$ can be decomposed into the span of $\phi(\boldsymbol{x}_i)$ and an orthogonal vector, that is,

$$\boldsymbol{w}=\sum_{i=1}^{n}\left(\alpha_i-\alpha_i^*\right)\phi\left(\boldsymbol{x}_i\right)+z=\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)+z,$$

where z satisfies $\left(\phi\left(\boldsymbol{x}_j\right)^T\cdot z\right)=0$ for all j, that is, $\boldsymbol{X}^Tz=\boldsymbol{0}$. Then we obtain the following equation:

$$\boldsymbol{X}^T\boldsymbol{w}=\boldsymbol{X}^T\left(\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)+z\right)=\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right).\tag{12}$$

According to Equation (12), the second and the third terms and the constraints of Formula (10) are independent of z. Besides, the last term of Formula (10) can also be considered as being independent of z. To simplify the first term of Formula (10), and consider $\boldsymbol{X}^Tz=\boldsymbol{0}$, we get

$$\begin{aligned}\boldsymbol{w}^T\boldsymbol{w}&=\left(\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)+z\right)^T\left(\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)+z\right)\\ &=(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+z^Tz\\ &\ge(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*),\end{aligned}$$

where the equal relationship in the above " $\ge''$ is valid if and only if $z=\boldsymbol{0}$. Thus, setting $z=\boldsymbol{0}$ does not affect the rest of the terms and strictly reduces the first term of Formula (10). Based on all above, $\boldsymbol{w}$ in Formula (10) can be represented as the form of Equation (11). **Q.E.D.**

Based on Theorem 1, we have

$$\boldsymbol{X}^T\boldsymbol{w}=\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)=\boldsymbol{Q}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right),$$

$$\boldsymbol{w}^T\boldsymbol{w}=(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right)=(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\boldsymbol{Q}\left(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*\right),$$

where $\boldsymbol{Q}=\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ is the kernel matrix. Let $\boldsymbol{\alpha}'=(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)$, thus Formula (9) leads to

$$\begin{aligned}\min_{\boldsymbol{\alpha}',\varepsilon,\boldsymbol{\xi},\boldsymbol{\xi}^*}\ & \frac{1}{2}(\boldsymbol{\alpha}')^T\boldsymbol{G}(\boldsymbol{\alpha}')+\boldsymbol{H}^T\boldsymbol{\alpha}'+C\left(v\varepsilon+\frac{1}{n}\left(\boldsymbol{e}^T\boldsymbol{\xi}+\boldsymbol{e}^T\boldsymbol{\xi}^*\right)\right)\\ s.t.\ & \boldsymbol{y}-\boldsymbol{Q}\boldsymbol{\alpha}'\le\varepsilon\boldsymbol{e}+\boldsymbol{\xi},\\ & \boldsymbol{Q}\boldsymbol{\alpha}'-\boldsymbol{y}\le\varepsilon\boldsymbol{e}+\boldsymbol{\xi}^*,\\ & \boldsymbol{\xi},\boldsymbol{\xi}^*\ge\boldsymbol{0},\\ & \varepsilon\boldsymbol{e}\ge\boldsymbol{0},\end{aligned}\tag{13}$$

where $\boldsymbol{G}=\boldsymbol{Q}+\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{Q}^T\boldsymbol{Q}-\frac{4\lambda_1}{n^2}\boldsymbol{Q}^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{Q}$ and $\boldsymbol{H}=-\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{Q}^T\boldsymbol{y}+\frac{4\lambda_1}{n^2}\boldsymbol{Q}^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y}$. By introducing the Lagrange multipliers $\boldsymbol{\eta}$, $\boldsymbol{\eta}^*$, $\boldsymbol{\beta}$, $\boldsymbol{\beta}^*$ and $\boldsymbol{\gamma}$, the Lagrange function of Formula (13) is given as follows:

$$\begin{aligned}&L(\boldsymbol{\alpha}',\boldsymbol{\xi},\boldsymbol{\xi}^*,\varepsilon,\boldsymbol{\beta},\boldsymbol{\beta}^*,\boldsymbol{\eta},\boldsymbol{\eta}^*,\boldsymbol{\gamma})\\ &=\frac{1}{2}(\boldsymbol{\alpha}')^T\boldsymbol{G}(\boldsymbol{\alpha}')+\boldsymbol{H}^T\boldsymbol{\alpha}'+C\left(v\varepsilon+\frac{1}{n}\left(\boldsymbol{e}^T\boldsymbol{\xi}+\boldsymbol{e}^T\boldsymbol{\xi}^*\right)\right)\\ &\quad-\boldsymbol{\beta}^T(\varepsilon\boldsymbol{e}+\boldsymbol{\xi}-\boldsymbol{y}+\boldsymbol{Q}\boldsymbol{\alpha}')\\ &\quad-\boldsymbol{\beta}^{*T}(\varepsilon\boldsymbol{e}+\boldsymbol{\xi}^*+\boldsymbol{y}-\boldsymbol{Q}\boldsymbol{\alpha}')-\boldsymbol{\eta}^T\boldsymbol{\xi}-\boldsymbol{\eta}^{*T}\boldsymbol{\xi}^*-\boldsymbol{\gamma}^T\varepsilon\boldsymbol{e},\end{aligned}\tag{14}$$

where $\boldsymbol{\beta}=[\beta_1,\beta_2,\ldots,\beta_n]^T$, $\boldsymbol{\beta}^*=[\beta_1^*,\beta_2^*,\ldots,\beta_n^*]^T$, $\boldsymbol{\eta}=[\eta_1,\eta_2,\ldots,\eta_n]^T$, $\boldsymbol{\eta}^*=[\eta_1^*,\eta_2^*,\ldots,\eta_n^*]^T$, and $\boldsymbol{\gamma}=[\gamma_1,\gamma_2,\ldots,\gamma_n]^T$. By setting the partial derivatives $\left\{\boldsymbol{\alpha}',\boldsymbol{\xi},\boldsymbol{\xi}^*,\varepsilon\right\}$ to zero for satisfying the KKT conditions [34], we can get the following equations:

$$\frac{\partial L}{\partial\boldsymbol{\alpha}'}=\boldsymbol{G}\boldsymbol{\alpha}'+\boldsymbol{H}-\boldsymbol{Q}^T\boldsymbol{\beta}+\boldsymbol{Q}^T\boldsymbol{\beta}^*=\boldsymbol{0},\tag{15}$$

$$\frac{\partial L}{\partial\boldsymbol{\xi}}=\frac{C}{n}\boldsymbol{e}-\boldsymbol{\beta}-\boldsymbol{\eta}=\boldsymbol{0},\tag{16}$$

$$\frac{\partial L}{\partial\boldsymbol{\xi}^*}=\frac{C}{n}\boldsymbol{e}-\boldsymbol{\beta}^*-\boldsymbol{\eta}^*=\boldsymbol{0},\tag{17}$$

$$\frac{\partial L}{\partial\varepsilon}=Cv-\boldsymbol{e}^T\boldsymbol{\beta}-\boldsymbol{e}^T\boldsymbol{\beta}^*-\boldsymbol{e}^T\boldsymbol{\gamma}=0.\tag{18}$$

By substituting Equations (15), (16), (17) and (18) into Equation (14), Equation (14) is re-written as:

$$\begin{aligned}\min_{\boldsymbol{\beta},\boldsymbol{\beta}^*}\ & f(\boldsymbol{\beta},\boldsymbol{\beta}^*)=\frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}^*)^T\boldsymbol{P}(\boldsymbol{\beta}-\boldsymbol{\beta}^*)+\boldsymbol{s}^T(\boldsymbol{\beta}-\boldsymbol{\beta}^*)\\ s.t.\ & \boldsymbol{e}^T\left(\boldsymbol{\beta}+\boldsymbol{\beta}^*\right)\le Cv,\\ & 0\le\beta_i,\beta_i^*\le\frac{C}{n},\quad i=1,2,\ldots,n,\end{aligned}\tag{19}$$

where $\boldsymbol{P}=\boldsymbol{Q}\boldsymbol{G}^{-1}\boldsymbol{Q}^T$ and $\boldsymbol{s}=-\boldsymbol{Q}\boldsymbol{G}^{-1}\boldsymbol{H}-\boldsymbol{y}$, $\boldsymbol{G}^{-1}$ stands for the inverse matrix of $\boldsymbol{G}$.

According to Chang and Lin, the inequality $\boldsymbol{e}^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)\le Cv$ in v-SVR can be replaced by the equality form of $\boldsymbol{e}^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)=Cv$ with the constraint $0<v\le1$, and there always exists the optimal solution [11]. Based on this conclusion, we can attain the equation for the following form:

$$\boldsymbol{e}^T\left(\boldsymbol{\beta}+\boldsymbol{\beta}^*\right)=Cv.\tag{20}$$

We thus substitute the equation $\boldsymbol{\beta}^* = Cv\boldsymbol{e} - \boldsymbol{\beta}$ into Formula (19), and Formula (19) can be obtained as follows:

$$\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \frac{1}{2}(2\boldsymbol{\beta} - Cv\boldsymbol{e})^T \boldsymbol{P}(2\boldsymbol{\beta} - Cv\boldsymbol{e}) + \boldsymbol{s}^T(2\boldsymbol{\beta} - Cv\boldsymbol{e})$$
$$s.t.\ 0 \le \beta_i \le \frac{C}{n}, \quad i = 1, 2, \ldots, n. \tag{21}$$

As one can see from Formula (21), by substituting the equation $\boldsymbol{\beta}^* = Cv\boldsymbol{e} - \boldsymbol{\beta}$ into Formula (19), the number of computational parameters of the v-MADR has been halved.

Due to the simple box constraint and the convex quadratic objective function, there exist many methods to solve the optimization problem [35]–[38]. To solve Formula (21), we use the DCD algorithm [39], which continuously selects one of the variables for minimization and keeps others as constants, thus a closed-form solution can be achieved at each iteration. In our situation, we minimize the variation of $f(\boldsymbol{\beta})$ by adjusting the value of $\beta_i \in \boldsymbol{\beta}$ with a step size of t while keeping other $\beta_{k\neq i}$ as constants, then we need to solve the following sub-problem:

$$\min_t f(\boldsymbol{\beta} + t\boldsymbol{d}_i)$$
$$s.t.\ 0 \le \beta_i + t \le \frac{C}{n}, \quad i = 1, 2, \ldots, n,$$

where $\boldsymbol{d}_i$ denotes the vector with 1 in the i-th element and $0's$ elsewhere. Thus, we have

$$f(\boldsymbol{\beta} + t\boldsymbol{d}_i) = f(\boldsymbol{\beta}) + [\nabla f(\boldsymbol{\beta})]_i t + 2p_{ii}t^2, \tag{22}$$

where p_{ii} is the diagonal entry of $\boldsymbol{P}$. Then we calculate the gradient $\nabla f(\boldsymbol{\beta})_i$ in Equation (22) as follows:

$$[\nabla f(\boldsymbol{\beta})]_i = 2\boldsymbol{d}_i^T \boldsymbol{P}(2\boldsymbol{\beta} - Cv\boldsymbol{e}) + 2\boldsymbol{s}^T\boldsymbol{d}_i.$$

As $f(\boldsymbol{\beta})$ is independent of t, it can be omitted from Equation (22). Hence $f(\boldsymbol{\beta} + t\boldsymbol{d}_i)$ can be transformed into a simple quadratic function. If we denote β_i^{iter} as the value of β_i at the $iter$-th iteration, $\beta_i^{iter+1} = \beta_i^{iter} + t$ is the value at the $(iter+1)$-th iteration. To solve Equation (22), we can have the minimization of t which satisfies Equation (22) for the following form:

$$t = -\frac{[\nabla f(\boldsymbol{\beta})]_i}{4p_{ii}}.$$

Thus, the value of β_i^{iter+1} is obtained as

$$\beta_i^{iter+1} = \beta_i^{iter} - \frac{[\nabla f(\boldsymbol{\beta})]_i}{4p_{ii}}.$$

Furthermore, considering the box constraint $0 \le \beta_i \le \frac{C}{n}$, we have the minimization for β_i^{iter+1} as follows:

$$\beta_i^{iter+1} \leftarrow min(max(\beta_i^{iter} - \frac{[\nabla f(\boldsymbol{\beta}^{iter})]_i}{4p_{ii}}, 0), C/n).$$

After $\boldsymbol{\beta}$ converges, we can obtain $\boldsymbol{\alpha}'$ according to Equation (15) and Equation (20) as follows:

$$\boldsymbol{\alpha}' = \boldsymbol{G}^{-1}\left(\boldsymbol{Q}^T(\boldsymbol{\beta} - \boldsymbol{\beta}^*) - \boldsymbol{H}\right) = \boldsymbol{G}^{-1}\left(\boldsymbol{Q}^T(2\boldsymbol{\beta} - Cv\boldsymbol{e}) - \boldsymbol{H}\right).$$

TABLE 1. Time complexity of the formula.

Formula being calculated	Time complexity of the formula
$\boldsymbol{Q} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$	$n * m * n$
$\boldsymbol{Q}_e = \boldsymbol{Q}^T\boldsymbol{e}$	n^2
$\boldsymbol{G}=\boldsymbol{Q}+\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{Q}^T\boldsymbol{Q} - \frac{4\lambda_1}{n^2}\boldsymbol{Q}^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{Q}$	$1 + n^3 + n^2$
$\boldsymbol{invG} = \boldsymbol{G}^{-1}$	n^3
$\boldsymbol{A} = \boldsymbol{G}^{-1}\boldsymbol{Q}$	n^3
$sumY = \boldsymbol{e}^T\boldsymbol{y}$	n
$\boldsymbol{H} = -(\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{Q}^T\boldsymbol{y} - \frac{4\lambda_1}{n^2}\boldsymbol{Q}^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y})$	n^2
$\boldsymbol{\alpha}' = \frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{G}^{-1}\boldsymbol{Q}^T\boldsymbol{y} - \frac{4\lambda_1}{n^2}\boldsymbol{G}^{-1}\boldsymbol{Q}^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y}$ $= -\boldsymbol{G}^{-1}\boldsymbol{H}$	n^2
$\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{G}^{-1}\boldsymbol{Q}^T = \boldsymbol{Q}\boldsymbol{A}$	n^3

Thus, the final function is

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i' \kappa(\boldsymbol{x}_i, \boldsymbol{x}),$$

where $\alpha_i' = (\alpha_i - \alpha_i^*)$.

Algorithm 1 summarizes the procedure of v-MADR with the kernel functions. The initial value of $\boldsymbol{\beta}$ is $Cv\boldsymbol{e}/2$, which simplifies the calculation procedure of v-MADR and satisfies Equation (20). Parameter v is controllable and its range is $(0, 1]$.

Algorithm 1 Dual Coordinate Descent Solver for Kernel v-MADR

Input: Dataset $\boldsymbol{X}$, λ_1, λ_2, C, v;
Output: $\boldsymbol{\alpha}'$;
Initialization: $\boldsymbol{\beta} = \frac{Cv\boldsymbol{e}}{2}, \boldsymbol{\alpha}' = \frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{G}^{-1}\boldsymbol{Q}^T\boldsymbol{y} - \frac{4\lambda_1}{n^2}\boldsymbol{G}^{-1}\boldsymbol{Q}^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y}, \boldsymbol{A} = \boldsymbol{G}^{-1}\boldsymbol{Q}^T, p_{ii} = \boldsymbol{d}_i^T\boldsymbol{Q}\boldsymbol{G}^{-1}\boldsymbol{Q}^T\boldsymbol{d}_i$;
1: **for** $iter = 1, 2, \ldots, maxIter$ **do**
2: Randomly disturb $\boldsymbol{\beta}$ and then get the random index;
3: **for** $i = 1, 2, \ldots, n$ **do**
4: $[\nabla f(\boldsymbol{\beta})]_i \leftarrow 2\left(\boldsymbol{d}_i^T\boldsymbol{Q}\boldsymbol{\alpha}' - y_i\right)$;
5: $\beta_i^{tmp} \leftarrow \beta_i$;
6: $\beta_i \leftarrow min(max(\beta_i - \frac{[\nabla f(\boldsymbol{\beta})]_i}{4p_{ii}}, 0), C/n)$;
7: $\boldsymbol{\alpha}' \leftarrow \boldsymbol{\alpha}' + 2\left(\beta_i - \beta_i^{tmp}\right)\boldsymbol{A}\boldsymbol{d}_i$;
8: **end for**
9: **if** $\boldsymbol{\beta}$ converges**then**
10: **break**;
11: **end if**
12: **end for**

We now analyze the computational complexity of Algorithm 1 as follows:

The parameters initialization is shown in Table 1, where n represents the number of the examples and m represents the number of features.

The time complexity for the dual coordinate descent (DCD) algorithm is $maxIter * n * n$, where $maxIter$ is 1000.

We can infer the time complexity of the DCD algorithm is the sum of the above time complexity. In summary, the time complexity of the DCD algorithm is $O\left(n^3\right)$ and it has the space complexity of $O\left(n^2\right)$.

2) LARGE-SCALE LINEAR KERNEL ν-MADR

In regression analysis, processing larger datasets may increase the time complexity. Although the DCD algorithm could solve kernel ν-MADR efficiently for small sample problems, it is not the best strategy for larger problems. Considering computational time cost, we adopt an averaged stochastic gradient descent (ASGD) algorithm [40] to linear kernel ν-MADR to improve the scalability of ν-MADR, and ASGD solves the optimization problem by computing a noisy unbiased estimate of the gradient, and it randomly samples a subset of the training instances rather than all data.

We reformulate Formula (10) into a linear kernel ν-MADR as follows:

$$\begin{aligned}\min_{\boldsymbol{w}}\ g(\boldsymbol{w}) = &\frac{1}{2}\boldsymbol{w}^T\left[\boldsymbol{I}+\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{X}\boldsymbol{X}^T-\frac{4\lambda_1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{X}^T\right]\boldsymbol{w}\\&+\left[-\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{X}\boldsymbol{y}+\frac{4\lambda_1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y}\right]^T\boldsymbol{w}\\&+\frac{C}{n}\left(\sum\nolimits_{i=1}^{n}\max(0,y_i-\boldsymbol{w}^T\boldsymbol{x}_i-\varepsilon)\right.\\&\left.+\sum\nolimits_{i=1}^{n}\max(0,\boldsymbol{w}^T\boldsymbol{x}_i-y_i-\varepsilon)\right),\end{aligned}\tag{23}$$

where $\boldsymbol{X}=[\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_n]$ and $\boldsymbol{y}=[y_1,y_2,\ldots,y_n]^T$. The term $Cv\varepsilon$ in Formula (10) is constant in an optimization problem, so we omit this term.

For large-scale problems, it is expensive to compute the gradient of Formula (23) because we need all the training samples for computation. Stochastic gradient descent (SGD) [41, 42] works by computing a noisy unbiased estimation of the gradient via sampling a subset of the training samples. When the objective is convex, the SGD is expected to converge to the global optimal solution. In recent years, SGD has been successfully used in various machine learning problems with powerful computation efficiency [43-46].

In order to obtain an unbiased estimation of the gradient $\nabla g(\boldsymbol{w})$, we first present the following theorem which can be proved by computing $\nabla g(\boldsymbol{w})$.

Theorem 2: If two samples $(\boldsymbol{x}_i,y_i)$ and $\left(\boldsymbol{x}_j,y_j\right)$ are sampled from the training data set randomly, then

$$\begin{aligned}\nabla g\left(\boldsymbol{w},\boldsymbol{x}_i,\boldsymbol{x}_j\right) = &\left(4\lambda_1+2\lambda_2\right)\boldsymbol{x}_i\boldsymbol{x}_i^T w-4\lambda_1 e_i\boldsymbol{x}_i e_j\boldsymbol{x}_j^T\boldsymbol{w}\\&+\boldsymbol{w}-\left(4\lambda_1+2\lambda_2\right)y_i\boldsymbol{x}_i+4\lambda_1 e_i\boldsymbol{x}_i e_j y_j\\&-C\begin{cases}\boldsymbol{x}_i & i\in I_1,\\-\boldsymbol{x}_i & i\in I_2,\\\boldsymbol{0} & otherwise,\end{cases}\end{aligned}\tag{24}$$

is an unbiased estimation of $\nabla g(\boldsymbol{w})$. Here $I_1=\left\{i\,|y_i-\boldsymbol{w}^T\boldsymbol{x}_i\geq\varepsilon\right\}$, $I_2=\left\{i\,|\boldsymbol{w}^T\boldsymbol{x}_i-y_i\geq\varepsilon\right\}$.

Proof: Note that the gradient of $g(\boldsymbol{w})$ is

$$\nabla g(\boldsymbol{w})=\boldsymbol{G}\boldsymbol{w}+\boldsymbol{H}-\frac{C}{n}\begin{cases}\sum_{i=1}^{n}\boldsymbol{x}_i & i\in I_1,\\\sum_{i=1}^{n}-\boldsymbol{x}_i & i\in I_2,\\\boldsymbol{0} & otherwise,\end{cases}$$

where $\boldsymbol{G}=\boldsymbol{I}+\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{X}\boldsymbol{X}^T-\frac{4\lambda_1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{X}^T$ and $\boldsymbol{H}=-\frac{4\lambda_1+2\lambda_2}{n}\boldsymbol{X}\boldsymbol{y}+\frac{4\lambda_1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y}$. Further note that

$$\begin{aligned}E_{\boldsymbol{x}_i}\left[\boldsymbol{x}_i\boldsymbol{x}_i^T\right]&=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^T=\frac{1}{n}\boldsymbol{X}\boldsymbol{X}^T,\\E_{\boldsymbol{x}_i}\left[y_i\boldsymbol{x}_i\right]&=\frac{1}{n}\sum_{i=1}^{n}y_i\boldsymbol{x}_i=\frac{1}{n}\boldsymbol{X}\boldsymbol{y},\\E_{\boldsymbol{x}_i}\left[e_i\boldsymbol{x}_i\right]&=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i=\frac{1}{n}\boldsymbol{X}\boldsymbol{e},\\E_{\boldsymbol{x}_i}\left[e_i y_i\right]&=\frac{1}{n}\sum_{i=1}^{n}y_i=\frac{1}{n}\boldsymbol{y}^T\boldsymbol{e}.\end{aligned}\tag{25}$$

According to the linearity of expectation, the independence between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and with the set of equations (25), we have

$$\begin{aligned}&E_{\boldsymbol{x}_i\boldsymbol{x}_j}\left[\nabla g\left(\boldsymbol{w},\boldsymbol{x}_i,\boldsymbol{x}_j\right)\right]\\&=\left(4\lambda_1+2\lambda_2\right)E_{\boldsymbol{x}_i}\left[\boldsymbol{x}_i\boldsymbol{x}_i^T\right]\boldsymbol{w}-4\lambda_1E_{\boldsymbol{x}_i}\left[e_i\boldsymbol{x}_i\right]E_{\boldsymbol{x}_j}\left[e_j\boldsymbol{x}_j\right]\boldsymbol{w}\\&\quad+\boldsymbol{w}-\left(4\lambda_1+2\lambda_2\right)E_{\boldsymbol{x}_i}\left[y_i\boldsymbol{x}_i\right]+4\lambda_1E_{\boldsymbol{x}_i}\left[e_i\boldsymbol{x}_i\right]E_{\boldsymbol{x}_j}\left[e_jy_j\right]\\&\quad-C\begin{cases}E_{\boldsymbol{x}_i} & [\boldsymbol{x}_i|i\in I_1],\\E_{\boldsymbol{x}_i} & [-\boldsymbol{x}_i|i\in I_2],\\\boldsymbol{0} & otherwise,\end{cases}\\&=\left(4\lambda_1+2\lambda_2\right)\frac{1}{n}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{w}-4\lambda_1\frac{1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{X}^T\boldsymbol{w}+\boldsymbol{w}\\&\quad-\left(4\lambda_1+2\lambda_2\right)\frac{1}{n}\boldsymbol{X}\boldsymbol{y}+4\lambda_1\frac{1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y}\\&\quad-C\frac{1}{n}\begin{cases}\sum_{i=1}^{n}\boldsymbol{x}_i, & i\in I_1,\\\sum_{i=1}^{n}-\boldsymbol{x}_i, & i\in I_2,\\\boldsymbol{0} & otherwise,\end{cases}\\&=\boldsymbol{G}\boldsymbol{w}+\boldsymbol{H}-\frac{C}{n}\begin{cases}\sum_{i=1}^{n}\boldsymbol{x}_i & i\in I_1,\\\sum_{i=1}^{n}-\boldsymbol{x}_i & i\in I_2,\\\boldsymbol{0} & otherwise,\end{cases}\\&=\nabla g(\boldsymbol{w}).\end{aligned}$$

It is shown that $\nabla g\left(\boldsymbol{w},\boldsymbol{x}_i,\boldsymbol{x}_j\right)$ is a noisy unbiased gradient of $g(\boldsymbol{w})$. **Q.E.D.**

Based on Theorem 2, the stochastic gradient can be updated as follows:

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t-\varphi_t\nabla g\left(\boldsymbol{w},\boldsymbol{x}_i,\boldsymbol{x}_j\right),\tag{26}$$

where φ_t is the learning rate at the t-th iteration.

Since the ASGD algorithm is more robust than the SGD algorithm [47], we actually adopt the ASGD algorithm to solve the optimization problem in Formula (23). At each iteration, in addition to updating the normal stochastic gradient in Equation (26), we also compute

$$\bar{\boldsymbol{w}}_t = \frac{1}{t - t_0} \sum_{i=t_0+1}^{t} \boldsymbol{w}_i,$$

where t_0 decides when to take the averaging operation. This average can also be calculated in a recursive formula as follows:

$$\bar{\boldsymbol{w}}_{t+1} = \bar{\boldsymbol{w}}_t + \delta_t \left(\boldsymbol{w}_{t+1} - \bar{\boldsymbol{w}}_t\right),$$

where $\delta_t = 1/max\{1, t - t_0\}$.

Algorithm 2 summarizes the procedure of large-scale linear kernel v-MADR.

Algorithm 2 Averaged Stochastic Gradient Descent Solver for Linear Kernel v-MADR

Input: Dataset $\boldsymbol{X}$, λ_1, λ_2, C, ε;
Output:;
Initialization: $\boldsymbol{u} = \boldsymbol{0}$, $t = 1, T = 5$;
1: **While** $t \leq T \cdot n$ **do**
2: Randomly select the training instances $\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right)$ and $\left(\boldsymbol{x}_j, \boldsymbol{y}_j\right)$;
3: Compute $\nabla g\left(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j\right)$ as in Equation (24);
4: $\boldsymbol{w} \leftarrow \varphi_t \nabla g\left(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{x}_j\right)$;
5: $\bar{\boldsymbol{w}} \leftarrow \bar{\boldsymbol{w}} + \delta_t \left(\boldsymbol{w} - \bar{\boldsymbol{w}}\right)$;
6: end while

The time complexity of the averaged stochastic gradient descent (ASGD) algorithm is $O\left(T * n * m\right)$ and its space complexity is $O\left(n * m\right)$.

3) PROPERTIES OF v-MADR

We study the statistical property of v-MADR that leads to a bound on the expectation of error for v-MADR according to the leave-one-out cross-validation estimate, which is an unbiased estimate of the probability of test error. For the sake of simplicity, we only discuss the linear case as shown Formula (10) here, in which $\boldsymbol{w}$ can be represented by the following form:

$$\boldsymbol{w} = \sum\nolimits_{i=1}^{n} \left(\alpha_i - \alpha_i^*\right) = \boldsymbol{\alpha} - \boldsymbol{\alpha}^*,$$

while the result is also used in kernel mapping situations ϕ. Then we can get the dual problem of Formula (10) using the same steps as in Section III.B.1, i.e.

$$\begin{aligned} &\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \frac{1}{2}(2\boldsymbol{\beta} - C\boldsymbol{v}\boldsymbol{e})^T \boldsymbol{P}(2\boldsymbol{\beta} - C\boldsymbol{v}\boldsymbol{e}) + \boldsymbol{s}^T(2\boldsymbol{\beta} - C\boldsymbol{v}\boldsymbol{e}) \\ &s.t.\ 0 \leq \beta_i \leq \frac{C}{n}, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{27}$$

where $\boldsymbol{P} = \boldsymbol{X}^T\boldsymbol{G}^{-1}\boldsymbol{X}$, $\boldsymbol{s} = -\boldsymbol{X}^T\boldsymbol{G}^{-1}\boldsymbol{H} - \boldsymbol{y}$, $\mathrm{G} = \frac{4\lambda_1 + 2\lambda_2}{n}\boldsymbol{X}\boldsymbol{X}^T - \frac{4\lambda_1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{X}^T + \boldsymbol{I}$ and $\boldsymbol{H} = -\frac{4\lambda_1 + 2\lambda_2}{n}\boldsymbol{X}\boldsymbol{y} + \frac{4\lambda_1}{n^2}\boldsymbol{X}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{y}$.

Definition 3: Regression error is defined as follows:

$$\pi\left(\boldsymbol{x}, y\right) = \left|y - f(\boldsymbol{x})\right|.$$

We give the following theorem to state the expectation of the probability of test error.

Theorem 3: Let $\boldsymbol{\beta}$ be the optimal solution of (27), and $E[R(\boldsymbol{\beta})]$ be the expectation of the probability of test error, then we have

$$E\left[R\left(\theta\right)\right] \leq \frac{E\left[\varepsilon\left|\boldsymbol{I}_1\right| + 2p\sum_{i\epsilon\boldsymbol{I}_2}\beta_i + \sum_{i\epsilon\boldsymbol{I}_3}\left(\varepsilon + \bar{\xi}_i\right)\right]}{n} \tag{28}$$

where $I_1 \equiv \{i|(\beta_i = 0) \cap (\beta_i^* = 0)\}$, $I_2 \equiv \{i|((0 < \beta_i < C/n) \cap (\beta_i^* = 0)) \cup ((0 < \beta_i^* < C/n) \cap (\beta_i = 0))\}$, $I_3 \equiv \{i|((\beta_i = C/n) \cap (\beta_i^* = 0)) \cup ((\beta_i^* = C/n) \cap (\beta_i = 0))\}$, $\bar{\xi}_i = max\{\xi_i, \xi_i^*\}$, $p = max\{p_{ii}, i = 0, 1, \cdots, n\}$, $\beta_i^* = Cv - \beta_i$ and p_{ii} is the diagonal entry of $\boldsymbol{P}$.

Proof: Suppose

$$\begin{aligned} \boldsymbol{\beta}^* &= \underset{0 \leq \boldsymbol{\beta} \leq \frac{c}{n}}{arg\,min}\ f(\boldsymbol{\beta}), \\ \boldsymbol{\beta}^{\boldsymbol{i}} &= \underset{0 \leq \boldsymbol{\beta} \leq \frac{c}{n}}{arg\,min}\ f(\boldsymbol{\beta}), \quad i = 1, 2, \cdots n \end{aligned}$$

and the corresponding solution for the linear kernel v-MADR are $\boldsymbol{w}'$ and $\boldsymbol{w}^i$, respectively.

According to [48],

$$E\left[R\left(\theta\right)\right] \leq \frac{E\left[L\left(\left(\boldsymbol{x}_1, y_1\right), \left(\boldsymbol{x}_2, y_2\right), \ldots, \left(\boldsymbol{x}_n, y_n\right)\right)\right]}{n} \tag{29}$$

where $L\left(\left(\boldsymbol{x}_1, y_1\right), \left(\boldsymbol{x}_2, y_2\right), \ldots c, \left(\boldsymbol{x}_n, y_n\right)\right)$ is the number of errors in the leave-one-out procedure.

In the process of solving Formula (27) using the Lagrange multipliers, every sample must meet the following KKT conditions:

$$\begin{aligned} &\beta_i\left(\varepsilon + \xi_i - y_i + \boldsymbol{x}_i^T\boldsymbol{\alpha}'\right) = 0, \\ &\beta_i^*\left(\varepsilon + \xi_i^* + y_i - \boldsymbol{x}_i^T\boldsymbol{\alpha}'\right) = 0, \\ &\left(\frac{C}{n} - \beta_i\right)\xi_i = 0, \\ &\left(\frac{C}{n} - \beta_i^*\right)\xi_i^* = 0, \\ &\xi_i, \xi_i^* \geq 0, i = 1, 2, \ldots, n, \\ &\varepsilon \geq 0. \end{aligned}$$

According to the KKT conditions, we have that if and only if $\varepsilon + \xi_i - y_i + \boldsymbol{x}_i^T\boldsymbol{\alpha}' = 0$, β_i can take a non-zero value, and if and only if $\varepsilon + \xi_i^* + y_i - \boldsymbol{x}_i^T\boldsymbol{\alpha}' = 0$, β_i^* can take a non-zero value. In other words, if the sample $(\boldsymbol{x}_i, y_i)$ is not in the ε-tube in the leave-one-out procedure, β_i and β_i^* can take a non-zero value. In addition, $\varepsilon + \xi_i - y_i + \boldsymbol{x}_i^T\boldsymbol{\alpha}' = 0$ and $\varepsilon + \xi_i^* + y_i - \boldsymbol{x}_i^T\boldsymbol{\alpha}' = 0$ cannot be established at the same time, so we get that at least one of β_i and β_i^* is zero. The specific breakdown is as follows:

i) If the sample $(\boldsymbol{x}_i, y_i)$ is in the ε-tube in the leave-one-out procedure, then $\varepsilon + \xi_i - y_i + \boldsymbol{x}_i^T\boldsymbol{\alpha}' \neq 0$ and $\varepsilon + \xi_i^* + y_i - \boldsymbol{x}_i^T\boldsymbol{\alpha}' \neq 0$, so we have $\beta_i = 0$ and $\beta_i^* = 0$;

ii) If the sample $(\boldsymbol{x}_i, y_i)$ is out of the ε-tube in the leave-one-out procedure, we have the following two situations:

a) if the sample is above the ε-tube, then $\xi_i \neq 0$ and $\varepsilon + \xi_i^* + y_i - \boldsymbol{x}_i^T \boldsymbol{\alpha}' \neq 0$. So we have $\beta_i = C/n$ and $\beta_i^* = 0$;

b) if the sample is under the ε-tube, then $\xi_i^* \neq 0$ and $\varepsilon + \xi_i - y_i + \boldsymbol{x}_i^T \boldsymbol{\alpha}' \neq 0$. So we have $\beta_i^* = C/n$ and $\beta_i = 0$;

iii) If the sample $(\boldsymbol{x}_i, y_i)$ is on the gap of the ε-tube in the leave-one-out procedure, we have the following two situations:

a) if the sample is on the upper gap of the ε-tube, then $\xi_i = 0$, and we have $0 < \beta_i \leq C/n$ and $\beta_i^* = 0$;

b) if the sample is on the lower gap of the ε-tube, then $\xi_i^* = 0$, and we have $0 < \beta_i^* \leq C/n$ and $\beta_i = 0$.

Based on the discussion above, we consider the following three cases to calculate the test error:

i) If both $\beta_i = 0$ and $\beta_i^* = 0$, we have that the sample $(\boldsymbol{x}_i, y_i)$ is in the ε-tube in the leave-one-out procedure, and $\pi(\boldsymbol{x}_i, y_i) \leq \varepsilon$.

ii) If $(0 < \beta_i < C/n) \cap (\beta_i^* = 0)$ or $(0 < \beta_i^* < C/n) \cap (\beta_i = 0)$, we have that

$$f\left(\boldsymbol{\beta}^{\boldsymbol{i}}\right) - f\left(\boldsymbol{\beta}^{\boldsymbol{i}} \boldsymbol{C}\, \imath \boldsymbol{d}_i\right) \leq f\left(\boldsymbol{\beta}^{\boldsymbol{i}}\right) - f\left(\boldsymbol{\beta}^{\boldsymbol{'}}\right), \quad (30)$$

$$f\left(\boldsymbol{\beta}^{\boldsymbol{i}}\right) - f\left(\boldsymbol{\beta}^{\boldsymbol{'}}\right) \leq f\left(\boldsymbol{\beta}^{\boldsymbol{'}} - \beta_i' \boldsymbol{d}_i\right) - f\left(\boldsymbol{\beta}^{\boldsymbol{'}}\right), \quad (31)$$

where $\boldsymbol{d}_i$ denotes the vector with 1 in the i-th element and $0's$ elsewhere. We can discovery that the left-hand side of formula (30) is equal to $[\nabla f(\boldsymbol{\beta})]_i^2 / (8p_{ii}) = \left(\boldsymbol{x}_i^T \boldsymbol{w}^i - y_i\right)^2 / (2p_{ii})$ and the right-hand side of formula (31) is equal to $2p_{ii}\beta_i'^2$. So by combining formula (30) and (31), we have $\pi(\boldsymbol{x}_i, y_i)^2 / (2p_{ii}) = \left(\boldsymbol{x}_i^T \boldsymbol{w}^i - y_i\right)^2 / (2p_{ii}) \leq 2p_{ii}\beta_i'^2$. Further, we can obtain $\pi(\boldsymbol{x}_i, y_i) \leq 2p_{ii}\beta_i'$.

iii) If $(\beta_i = C/n) \cap (\beta_i^* = 0)$ or $(\beta_i^* = C/n) \cap (\beta_i = 0)$, we have that the sample $(\boldsymbol{x}_i, y_i)$ is not in the ε-tube in the leave-one-out procedure. So we can get $\pi(\boldsymbol{x}_i, y_i) = \varepsilon + \bar{\xi}_i'$, where $\bar{\xi}_i' = max\xi_i', \xi_i^{*\prime}$.

So we have

$$L((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)) \leq \varepsilon |\boldsymbol{I}_1| + 2p \sum\nolimits_{i \epsilon \boldsymbol{I}_2} \beta_i' + \sum\nolimits_{i \epsilon \boldsymbol{I}_3} \left(\varepsilon + \bar{\xi}_i'\right),$$

where $I_1 \equiv \{i | (\beta_i' = 0) \cap (\beta_i^{*\prime} = 0)\}$, $I_2 \equiv \{i | ((0 < \beta_i' < C/n) \cap (\beta_i^{*\prime} = 0)) \cup ((0 < \beta_i^{*\prime} < C/n) \cap (\beta_i' = 0))\}$, $I_3 \equiv \{i | ((\beta_i' = C/n) \cap (\beta_i^{*\prime} = 0)) \cup ((\beta_i^{*\prime} = C/n) \cap (\beta_i' = 0))\}$, $\bar{\xi}_i' = max\xi_i', \xi_i^{*\prime}, p = max\{p_{ii}, i = 0, 1, \cdots, n\}$ and $\beta_i^{*\prime} = Cv - \beta_i'$. Take expectation on both side and with formula (29), we reach the conclusion that formula (28) holds. **Q.E.D.**

IV. EXPERIMENTAL RESULTS

Since SVR-based algorithms are now widely used for regression problems and demonstrate better generalization ability [8]–[10] than many existing algorithms, such as least square regression [4], Neural Networks (NN) regression [5], logistic regression [6], and ridge regression [7], we will not repeat these comparisons. In this section, we empirically evaluate the performance of our ν-MADR compared with other SVR-based algorithms, including ν-SVR, LS-SVR, ε-TSVR, and linear ε-SVR on several datasets, including two artificial datasets, eight medium-scale datasets, and six large-scale datasets. All algorithms are implemented with MATLAB R2014a on a PC with a 2.00GHz CPU and 32 GB memory. ε-SVR is solved by LIBSVM [49]; ε-SVR is solved by LIBLINEAR [50]; LS-SVR is solved by LSSVMlab [51]; and ε-TSVR is solved by the SOR technique [52], [53]. RBF kernel $\kappa\left(\boldsymbol{x}_i^T, \boldsymbol{x}_j^T\right) = \exp\left(-\left\|\boldsymbol{x}_i^T - \boldsymbol{x}_j^T\right\|^2 / \sigma^2\right)$ and polynomial kernel $\kappa\left(\boldsymbol{x}_i^T, \boldsymbol{x}_j^T\right) = \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1\right)^d$ are employed for nonlinear regression. The values of the parameters are obtained by means of a grid-search method [54]. For brevity, we set $c_1 = c_2, c_3 = c_4$ and $\varepsilon_1 = \varepsilon_2$ for ε-TSVR and $\lambda_1 = \lambda_2$ for our nonlinear ν-MADR. The parameter ν in ν-MADR is selected from the set $\left\{2^{-9}, 2^{-8}, \ldots, 2^0\right\}$, and the remaining parameters in the five methods and the parameters in the Gaussian kernel are selected from the set $\left\{2^{-9}, 2^{-8}, \ldots, 2^9\right\}$ by 10-fold cross-validation. Specifically, the parameter d in polynomial kernel is selected from {2, 3, 4, 5, 6}.

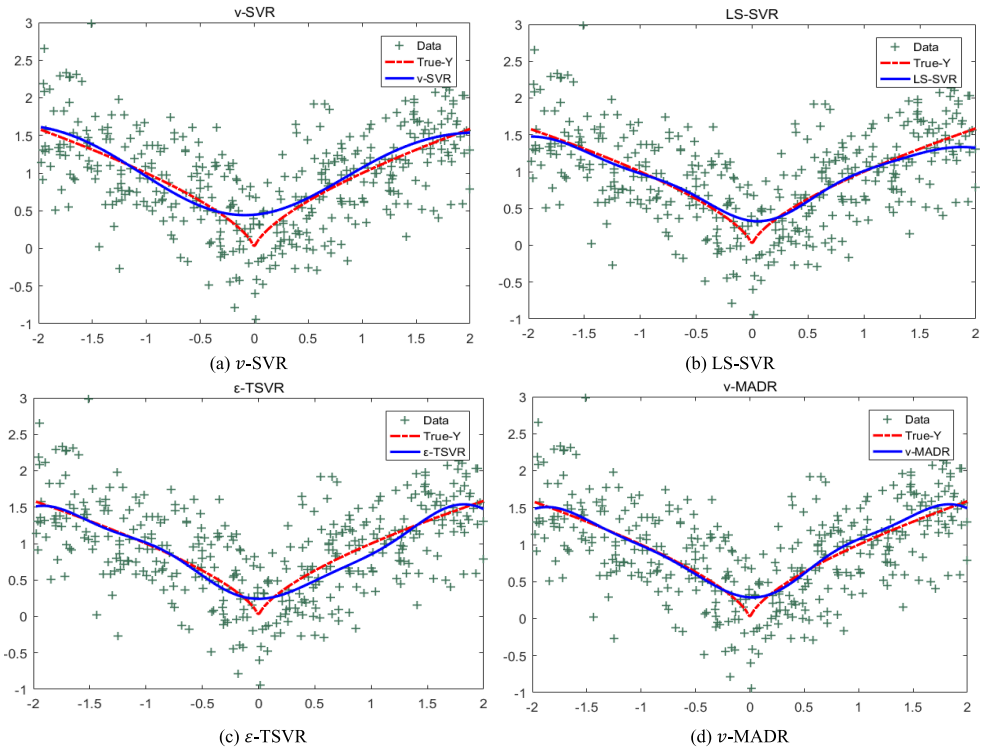
TABLE 2. Performance metrics.

Metrics	Definition
SSE	$\mathrm{SSE} = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$
SST	$\mathrm{SST} = \sum_{i=1}^{m} (y_i - \bar{y})^2$
SSR	$\mathrm{SSR} = \sum_{i=1}^{m} (\hat{y}_i - \bar{y})^2$
NMSE	$\mathrm{NMSE} = \mathrm{SSE}/\mathrm{SST}$
R^2	$\mathrm{R}^2 = \frac{\left(\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - E[\hat{y}_i])(y_i - E[y_i])\right)^2}{\sigma_{\hat{y}}^2 \sigma_{\hat{y}}^2}$
MAPE	$\mathrm{MAPE} = \frac{1}{m}\sum_{i=1}^{m}\left\|\frac{y_i - \hat{y}_i}{y_i}\right\|$

In order to evaluate the performance of the proposed algorithm, the performance metrics are specified before presenting the experimental results. Without loss of generality, let n be the number of training samples and m be the number of testing samples, denote $\hat{y}_i$ as the prediction value of y_i, and $\bar{y} = \left(\sum_{i=1}^{m} y_i\right)/m$ as the average value of $y_1, y_2, \ldots, y_m$. Then the details of the metrics used for assessing the performance of all regression algorithms are stated in Table 2. To demonstrate the overall performance of a method, a performance metric referred to average rank of each method is

TABLE 3. The real-world datasets used for experiments.

Scale	Dataset	Samples	Features	Dataset	Samples	Features
medium	Diabetes	43	2	Motorcycle	133	1
	Autoprice	159	15	Servo	167	4
	Wisconsin	194	32	MachineCPU	209	31
	AutoMpg	398	7	WDBC	569	30
large	ConcreteCS	1030	8	Abalone	4177	8
	CPUsmall	8192	12	Bike	10886	9
	Driftdataset	13910	128	Cadate	20640	8



(a) v-SVR (b) LS-SVR (c) ε-TSVR (d) v-MADR

FIGURE 1. The predictions of v-SVR, LS-SVR, ε-TSVR and our v-MADR on function $y = x^{\frac{2}{3}}$.

defined as

$$average\ rank\ (R) = \frac{1}{s}\sum_{i=1}^{s} rank(R)_i,$$

where $R \in \{v-\text{SVR}, \text{LS}-\text{SVR}, \varepsilon-\text{TSVR}, \text{LIBLINE-AR}, v-\text{MADR}\}$ is the regression method, s is the number of datasets, and $\text{rank}(R)_i$ means the performance rank of method R on the i-th dataset among all regression methods.

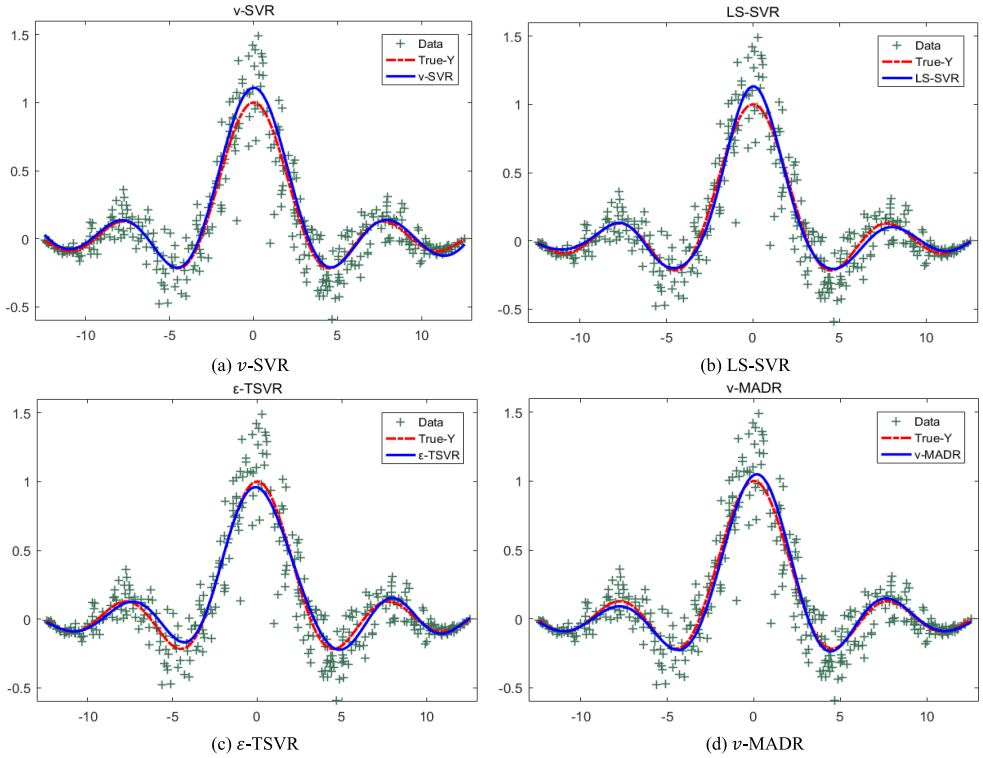
In our experiments, we test the performance of the above methods on two artificial datasets, eight medium-scale datasets and six large-scale data sets. The basic information of these datasets is given in Table 3. All real-world datasets are taken from UCI (http://archive.ics.uci.edu/ml) and StatLib (http:// lib.stat.cmu.edu/), and more detailed information can be accessed from those websites. Before regression analysis, all of these real datasets are normalized to zero mean and unit deviation. For medium-scale datasets, RBF kernel and polynomial kernel are used, and for large-scale datasets, only the linear kernel v-MADR is used considering the computational complexity. Each experiment is repeated for 30 trials with 10-fold cross validation and the mean evaluation of R^2, NMSE, MAPE and their standard deviations were recorded. Particularly, the two datasets "Diabetes" and "Motorcycle" have smaller numbers of samples and features, so we use the leave-one-out cross validation instead.

V. ARTIFICIAL DATASETS

In order to compare our v-MADR with v-SVR, LS-SVR, and ε-TSVR, we choose two artificial datasets with different distributions. Firstly, we consider the function: $y = x^{\frac{2}{3}}$. In order to fully assess the performance of the methods, the training samples are added with Gaussian noises with zero means and 0.5 standard deviation, that is, we have the following training samples (x_i, y_i):

$$y_i = x_i^{\frac{2}{3}} + \xi_i, x_i \sim U\left[-2, 2\right], \xi_i \sim N\left(0, 0.5^2\right), \quad (32)$$

where $U\left[a, b\right]$ represents the uniformly random variable in $[a, b]$ and $N\left(\mu, \sigma^2\right)$ represents the Gaussian random variable

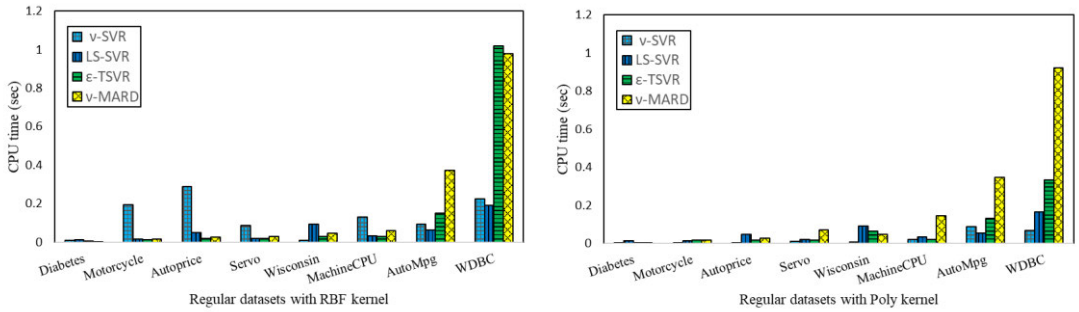


(a) ν-SVR (b) LS-SVR (c) ε-TSVR (d) ν-MADR

FIGURE 2. The predictions of ν-SVR, LS-SVR, ε-TSVR and our ν-MADR on the sinc function.

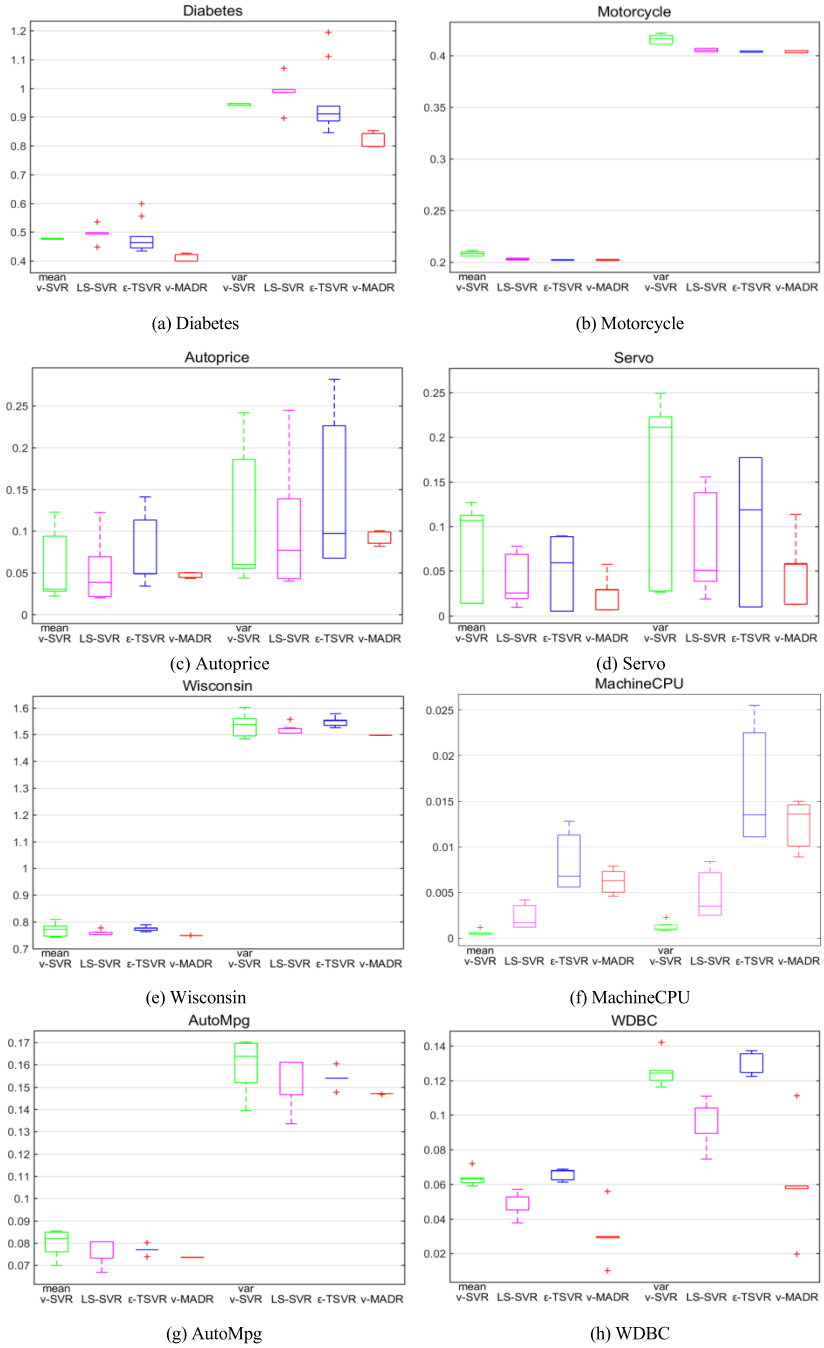
TABLE 4. The result comparisons of ν-SVR, LS-SVR, ε-TSVR and our ν-MADR on artificial datasets.

Dataset	regressor	R^2 (rank)	NMSE (rank)	MAPE (rank)	CPU(sec)
$x^{\frac{2}{3}}$	ν-SVR	0.9319(4)	0.0856(4)	0.2479(4)	0.0333
	LS-SVR	0.9446(3)	0.0698(3)	0.2152(2)	0.0188
	ε-TSVR	0.9500(2)	0.0593(2)	**0.2091(1)**	0.0196
	ν-MADR	**0.9584(1)**	**0.0529(1)**	0.2165(3)	0.0471
$Sinc(x)$	ν-SVR	0.9889(2)	0.0183(2)	1.1792(4)	0.0837
	LS-SVR	0.9844(3)	0.0190(3)	0.8389(2)	0.0172
	ε-TSVR	0.9823(4)	0.0200(4)	0.9118(3)	0.0202
	ν-MADR	**0.9940(1)**	**0.0083(1)**	**0.7333(1)**	0.0474
average rank	ν-SVR	3	3	4	-
	LS-SVR	3	3	2	-
	ε-TSVR	3	3	2	-
	ν-MADR	**1**	**1**	**2**	-



(a) CPU time on medium-scale datasets with RBF kernel (b) CPU time on medium-scale datasets with polynomial kernel

FIGURE 3. The CPU time on medium-scale datasets.



(a) Diabetes (b) Motorcycle

(c) Autoprice (d) Servo

(e) Wisconsin (f) MachineCPU

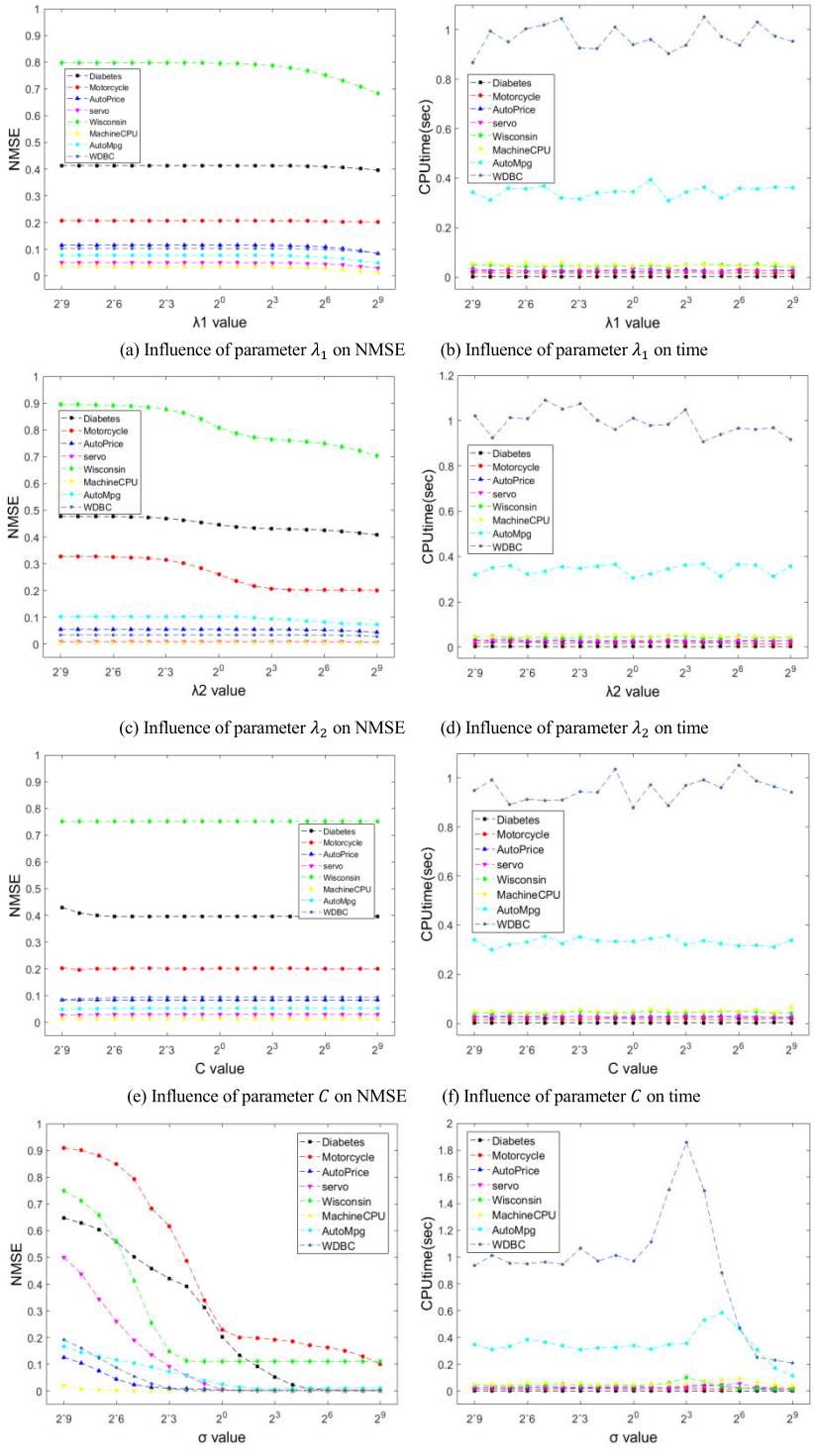
(g) AutoMpg (h) WDBC

FIGURE 4. The absolute regression deviation mean and variance of v-MADR with RBF kernel on medium-scale datasets.

with means μ and standard deviation σ, respectively. To avoid biased comparisons, ten independent groups of noisy samples are randomly generated, including 200 training samples and 400 none noise test samples.

The estimated functions obtained by these four methods are shown in Figure 1. Obviously, all four methods have obtained good fitted values, but our v-MADR has the best approximation compared to the rest of the methods. Table 4 shows the corresponding performance metrics and training time. Compared with the other methods, our v-MADR has the highest R^2, lowest NMSE and MAPE, which indicates that our v-MADR achieves good fitting performance and a good presentation of the statistical information in the training dataset. In addition, the CPU time of our v-MADR is not much different from other methods, and equivalent to v-SVR.

The second artificial example is the regression estimation on the *Sinc* function: $y = sin(x)/x$. The training samples are



(a) Influence of parameter λ_1 on NMSE (b) Influence of parameter λ_1 on time

(c) Influence of parameter λ_2 on NMSE (d) Influence of parameter λ_2 on time

(e) Influence of parameter C on NMSE (f) Influence of parameter C on time

(g) Influence of parameter σ on NMSE (h) Influence of parameter σ on time

FIGURE 5. Parameter influence on NMSE and CPU time on medium-scale datasets with RBF kernel.

added with Gaussian noise with zero means and 0.5 standard deviation. Therefore, we have the following training samples (x_i, y_i):

$$y_i = \frac{\sin(x_i)}{x_i} + \left(0.5 - \frac{|x_i|}{8\pi}\right)\xi_i,$$

$$x_i \sim U[-4\pi, 4\pi], \quad \xi_i \sim N(0, 0.5^2). \tag{33}$$

The dataset consists of 200 training samples and 400 test samples. Figure 2 illustrates the estimated functions obtained by these four methods and Table 4 shows the corresponding performance. These results also demonstrate the superiority

TABLE 5. The result comparisons of v-SVR, LS-SVR, ε-TSVR and v-MADR on medium-scale datasets with RBF kernel.

Dataset	regressor	R^2 (rank)	NMSE (rank)	MAPE (rank)	CPU(sec)
Diabetes	v-SVR	0.5343 ± 0.0028(2)	0.4768 ± 0.0024(2)	1.5971 ± 0.0197(4)	0.009
	LS-SVR	0.5151 ± 0.0381(4)	0.4986 ± 0.0504(4)	1.5742 ± 0.0477(2)	0.014
	ε-TSVR	0.5281 ± 0.1182(3)	0.4805 ± 0.1185(3)	1.5909 ± 0.1188(3)	0.004
	v-MADR	**0.5891 ± 0.0137(1)**	**0.4127 ± 0.0141(1)**	**1.4750 ± 0.0493(1)**	0.002
Motorcycle	v-SVR	0.7938 ± 0.0020(3)	0.2081 ± 0.0029(4)	1.2636 ± 0.0352(4)	0.007
	LS-SVR	0.7975 ± 0.0011(2)	0.2027 ± 0.0009(3)	1.2444 ± 0.0154(3)	0.013
	ε-TSVR	0.7680 ± 0.0003(4)	0.2021 ± 0.0003(2)	1.2441 ± 0.0055(2)	0.012
	v-MADR	**0.7984 ± 0.0006(1)**	**0.2017 ± 0.0007(1)**	**1.2316 ± 0.0063(1)**	0.016
Autoprice	v-SVR	0.9481 ± 0.0680(2)	0.0530 ± 0.0698(2)	**0.4187 ± 0.1070(1)**	0.004
	LS-SVR	0.9465 ± 0.0674(3)	0.0541 ± 0.0682(3)	0.5039 ± 0.1392(3)	0.049
	ε-TSVR	0.9338 ± 0.0744(4)	0.0665 ± 0.0745(4)	0.5355 ± 0.1524(4)	0.007
	v-MADR	**0.9549 ± 0.0040(1)**	**0.0467 ± 0.0035(1)**	0.4889 ± 0.0220(2)	0.027
Servo	v-SVR	0.9337 ± 0.0565(4)	0.0686 ± 0.0582(4)	0.2491 ± 0.0420(2)	0.291
	LS-SVR	0.9630 ± 0.0403(2)	0.0372 ± 0.0406(2)	0.3374 ± 0.1835(3)	0.019
	ε-TSVR	0.9507 ± 0.0445(3)	0.0495 ± 0.0446(3)	0.3931 ± 0.2082(4)	0.016
	v-MADR	**0.9755 ± 0.0187(1)**	**0.0253 ± 0.0320(1)**	**0.2473 ± 0.1280(1)**	0.045
Wisconsin	v-SVR	0.2420 ± 0.0354(3)	0.7712 ± 0.0371(3)	1.8255 ± 0.1265(4)	0.008
	LS-SVR	0.2546 ± 0.0146(2)	0.7574 ± 0.0202(2)	**1.5667 ± 0.0462(1)**	0.086
	ε-TSVR	0.2285 ± 0.0130(4)	0.7752 ± 0.0137(4)	1.7028 ± 0.1055(3)	0.027
	v-MADR	**0.2641 ± 0.0016(1)**	**0.7486 ± 0.0012(1)**	1.5840 ± 0.0125(2)	0.046
MachineCPU	v-SVR	**0.9994 ± 0.0005(1)**	**0.0006 ± 0.0006(1)**	**0.0888 ± 0.1165(1)**	0.006
	LS-SVR	0.9978 ± 0.0020(2)	0.0022 ± 0.0020(2)	0.1775 ± 0.1113(2)	0.027
	ε-TSVR	0.9921 ± 0.0048(4)	0.0080 ± 0.0048(4)	0.3580 ± 0.1641(4)	0.034
	v-MADR	0.9942 ± 0.0014(3)	0.0063 ± 0.0017(3)	0.2134 ± 0.0779(3)	0.047
AutoMpg	v-SVR	0.9196 ± 0.0108(4)	0.0807 ± 0.0121(4)	1.0111 ± 0.2842(2)	0.118
	LS-SVR	0.9262 ± 0.0072(2)	0.0741 ± 0.0079(2)	1.0196 ± 0.0977(3)	0.054
	ε-TSVR	0.9228 ± 0.0034(3)	0.0773 ± 0.0035(3)	1.0549 ± 0.0421(4)	0.105
	v-MADR	**0.9267 ± 0.0017(1)**	**0.0736 ± 0.0015(1)**	**1.0080 ± 0.0182(1)**	0.326
WDBC	v-SVR	0.9382 ± 0.0106(3)	0.0632 ± 0.0086(3)	0.0988 ± 0.0041(3)	0.216
	LS-SVR	0.9520 ± 0.0109(2)	0.0489 ± 0.0113(2)	0.0182 ± 0.0052(1)	0.196
	ε-TSVR	0.9344 ± 0.0066(4)	0.0659 ± 0.0047(4)	0.1726 ± 0.0085(4)	0.615
	v-MADR	**0.9710 ± 0.0258(1)**	**0.0298 ± 0.0198(1)**	**0.0714 ± 0.0043(2)**	0.913
average rank	v-SVR	2.7500	2.8750	2.6250	-
	LS-SVR	2.3750	2.5000	2.2500	-
	ε-TSVR	3.6250	3.3750	3.5000	-
	v-MADR	**1.2500**	**1.2500**	**1.6250**	-

of our method. At the bottom of Table 4, we list the average ranks of all four methods on the artificial datasets for different performance metrics. It can be seen that our v-MADR is superior to other three methods on R^2 and NMSE, and is comparable to LS-SVR and ε-TSVR in terms of MAPE.

A. MEDIUM-SCALE DATASETS

Table 5 and Table 6 list the experimental results on the eight medium-scale datasets from UCI and StatLib with RBF and polynomial kernels, respectively. From the average rank at the bottom of Table 5 and Table 6, our v-MADR is superior to the other three methods. In detail, on most datasets, our v-MADR has the highest R^2, lowest NMSE and MAPE. Although on several datasets, such as "MachineCPU", our v-MADR does not achieve the best experimental results compared with other methods, it is not the worst. Our v-MADR also has good performance in terms of CPU running time. The above experimental results indicate that v-MADR is an efficient and promising algorithm for regression. Table 7 and Table 8 list the optimal parameters with RBF and polynomial kernels, respectively. Figure 3(a) and Figure 3(b) show the comparisons of CPU time among our v-MADR, v-SVR, LS-SVR and ε-TSVR on each medium-scale dataset with RBF kernel and polynomial kernel.

TABLE 6. The result comparisons of v-SVR, LS-SVR, ε-TSVR and v-MADR on medium-scale datasets with polynomial kernel.

Dataset	regressor	R^2 (rank)	NMSE (rank)	MAPE (rank)	CPU(sec)
Diabetes	v-SVR	0.371 ± 0.0931(4)	0.6424 ± 0.1007(4)	1.2135 ± 0.5799(2)	0.001
	LS-SVR	0.526 ± 0.0096(2)	0.4901 ± 0.0313(3)	1.4929 ± 0.1274(3)	0.012
	ε-TSVR	0.525 ± 0.0063(3)	0.4846 ± 0.0214(2)	1.6795 ± 0.1074(4)	0.004
	v-MADR	**0.549 ± 0.0324(1)**	**0.4834 ± 0.0317(1)**	**1.1254 ± 0.0465(1)**	0.002
Motorcycle	v-SVR	0.115 ± 0.0106(4)	0.8898 ± 0.0103(4)	**1.2730 ± 0.1945(1)**	0.001
	LS-SVR	0.547 ± 0.0024(3)	0.4544 ± 0.0055(3)	1.6929 ± 0.0356 (4)	0.014
	ε-TSVR	0.548 ± 0.0010(2)	0.4540 ± 0.0042(2)	1.6771 ± 0.0540 (3)	0.032
	v-MADR	**0.549 ± 0.0006(1)**	**0.4516 ± 0.0028(1)**	1.6456 ± 0.0315 (2)	0.017
Autoprice	v-SVR	0.881 ± 0.0433(4)	0.1217 ± 0.0551(4)	0.6501 ± 0.1275(4)	0.004
	LS-SVR	0.965 ± 0.0109(3)	0.0349 ± 0.0113(3)	0.4184 ± 0.0446(2)	0.043
	ε-TSVR	0.973 ± 0.0082(2)	0.0315 ± 0.0152(2)	0.4750 ± 0.1303(3)	0.017
	v-MADR	**0.976 ± 0.0108(1)**	**0.0256 ± 0.0115(1)**	**0.4056 ± 0.0772(1)**	0.025
Servo	v-SVR	0.545 ± 0.0034(4)	0.4905 ± 0.0079(4)	0.8068 ± 0.0122(4)	0.016
	LS-SVR	0.935 ± 0.0385(3)	0.0645 ± 0.0384(3)	0.4870 ± 0.1473(3)	0.020
	ε-TSVR	0.942 ± 0.0012(2)	**0.0576 ± 0.0017(1)**	0.4856 ± 0.0112(2)	0.019
	v-MADR	**0.945 ± 0.0047(1)**	0.0631 ± 0.0066(2)	**0.4425 ± 0.1647(1)**	0.083
Wisconsin	v-SVR	0.203 ± 0.0227(4)	0.8326 ± 0.0357(4)	1.3011 ± 0.0160(2)	0.007
	LS-SVR	0.469 ± 0.0021(3)	0.5692 ± 0.0092(3)	1.3614 ± 0.0036(3)	0.095
	ε-TSVR	**0.777 ± 0.0020(1)**	**0.2439 ± 0.0336(1)**	1.3793 ± 0.1926(4)	0.039
	v-MADR	0.585 ± 0.0013(2)	0.4577 ± 0.0066(2)	**0.8205 ± 0.0081(1)**	0.041
MachineCPU	v-SVR	0.933 ± 0.0175(4)	0.0682 ± 0.0171(4)	1.3525 ± 0.2995(4)	0.015
	LS-SVR	0.999 ± 0.0008(3)	0.0006 ± 0.0009(3)	0.1068 ± 0.0696(2)	0.029
	ε-TSVR	0.999 ± 0.0004(2)	0.0005 ± 0.0005(2)	0.1159 ± 0.0503(3)	0.021
	v-MADR	**0.999 ± 0.0002(1)**	**0.0004 ± 0.0003(1)**	**0.0916 ± 0.0251(1)**	0.045
AutoMpg	v-SVR	0.802 ± 0.0127(4)	0.1997 ± 0.0136(4)	1.2530 ± 0.2663(2)	0.080
	LS-SVR	0.895 ± 0.0307(2)	0.1047 ± 0.0309(2)	1.2613 ± 0.2146(3)	0.047
	ε-TSVR	0.892 ± 0.0396(3)	0.1079 ± 0.0356(3)	1.3668 ± 0.0156(4)	0.143
	v-MADR	**0.922 ± 0.0131(1)**	**0.0815 ± 0.0154(1)**	**1.1219 ± 0.2950(1)**	0.298
WDBC	v-SVR	0.471 ± 0.0198(4)	0.5494 ± 0.0537(4)	0.7195 ± 0.0166(4)	0.061
	LS-SVR	0.921 ± 0.0202(2)	0.0812 ± 0.0211(2)	**0.0318 ± 0.0094(1)**	0.177
	ε-TSVR	0.902 ± 0.0132(3)	0.0977 ± 0.0124(3)	0.2362 ± 0.0198(3)	0.360
	v-MADR	**0.980 ± 0.0157(1)**	**0.0202 ± 0.0062(1)**	0.0510 ± 0.0251(2)	1.112
average rank	v-SVR	4.0000	4.0000	2.8750	-
	LS-SVR	2.6250	2.7500	2.6250	-
	ε-TSVR	2.2500	2.0000	3.2500	-
	v-MADR	**1.1250**	**1.2500**	**1.2500**	-

TABLE 7. The optimal parameters on medium-scale datasets with RBF kernel.

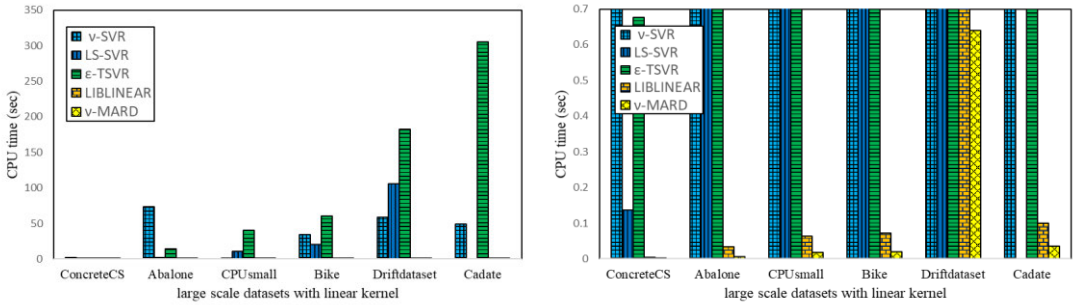
Dataset	v-SVR		LS-SVR		ε-TSVR		v-MADR			
	C	σ	C	σ	$c_1 = c_2$	$c_3 = c_4$	σ	C	$\lambda_1 = \lambda_2$	σ
Diabetes	2^9	2^{-4}	2^7	2^4	2^{-4}	2^{-3}	2^{-2}	2^5	2^7	2^{-3}
Motorcycle	2^9	2^0	2^4	2^{-2}	2^{-8}	2^{-6}	2^1	2^9	2^9	2^1
Autoprice	2^9	2^{-8}	2^8	2^{-6}	2^{-6}	2^{-9}	2^{-6}	2^{-9}	2^9	2^{-6}
Servo	2^6	2^{-1}	2^9	2^1	2^{-9}	2^{-9}	2^{-2}	2^{-7}	2^9	2^0
Wisconsin	2^2	2^{-9}	2^2	2^9	2^{-5}	2^2	2^{-7}	2^{-1}	2^6	2^{-9}
MachineCPU	2^8	2^{-9}	2^9	2^8	2^{-7}	2^{-9}	2^{-8}	2^{-9}	2^9	2^{-7}
AutoMpg	2^3	2^{-3}	2^4	2^2	2^{-5}	2^{-7}	2^{-3}	2^{-5}	2^9	2^{-3}
WDBC	2^2	2^{-5}	2^3	2^4	2^{-5}	2^{-6}	2^{-5}	2^{-9}	2^9	2^{-4}

TABLE 8. The optimal parameters on medium-scale datasets with polynomial kernel.

Dataset	ν-SVR		LS-SVR		ε-TSVR			ν-MADR		
	C	d	C	d	$c_1 = c_2$	$c_3 = c_4$	d	C	$\lambda_1 = \lambda_2$	d
Diabetes	2^{-1}	3	2^{-4}	2	2^{-3}	2^{8}	2	2^{9}	2^{-5}	4
Motorcycle	2^{-3}	2	2^{-3}	6	2^{-5}	2^{7}	6	2^{-4}	2^{4}	6
Autoprice	2^{-1}	3	2^{-5}	2	2^{-3}	2^{3}	2	2^{-1}	2^{3}	2
Servo	2^{2}	3	2^{-6}	5	2^{-9}	2^{5}	4	2^{-3}	2^{-4}	5
Wisconsin	2^{-3}	3	2^{-9}	2	2^{-1}	2^{9}	2	2^{9}	2^{-9}	3
MachineCPU	2^{0}	2	2^{-2}	2	2^{-8}	2^{-1}	2	2^{-3}	2^{-9}	2
AutoMpg	2^{2}	3	2^{-4}	3	2^{-2}	2^{9}	2	2^{-3}	2^{2}	3
WDBC	2^{-3}	3	2^{-7}	2	2^{-5}	2^{4}	2	2^{9}	2^{-3}	4

TABLE 9. The result comparisons of ν-SVR, LS-SVR, ε-TSVR, ε-SVR and ν-MADR on large-scale datasets with linear kernel.

Dataset	regressor	R^2 (rank)	NMSE (rank)	MAPE (rank)	CPU(sec)
ConcreteCS	ν-SVR	0.5849 ± 0.0511(4)	0.4180 ± 0.0535(4)	2.1691 ± 0.4274(4)	2.1470
	LS-SVR	0.6030 ± 0.0440(2)	0.4033 ± 0.0504(2)	2.0463 ± 0.6259(2)	0.1371
	ε-TSVR	0.5934 ± 0.0483(3)	0.4076 ± 0.0476(3)	2.0746 ± 0.5095(3)	0.6757
	LIBLINEAR	0.3945 ± 0.0493(5)	0.7194 ± 0.1146(5)	3.9093 ± 2.1885(5)	0.0040
	ν-MADR	**0.6124 ± 0.0251(1)**	**0.3925 ± 0.0278(1)**	**1.8771 ± 0.6436(1)**	0.0024
Abalone	ν-SVR	0.5237 ± 0.0399(3)	0.4838 ± 0.0403(3)	3.3437 ± 0.3106(2)	73.7317
	LS-SVR	0.5103 ± 0.0345(4)	0.4921 ± 0.0351(4)	3.4757 ± 0.4091(5)	1.7781
	ε-TSVR	0.5310 ± 0.0267(2)	0.4712 ± 0.0286(2)	3.4732 ± 0.2775(4)	14.2850
	LIBLINEAR	0.3605 ± 0.0295(5)	0.6708 ± 0.0586(5)	3.3933 ± 0.6411(3)	0.0337
	ν-MADR	**0.5491 ± 0.0322(1)**	**0.4569 ± 0.0370(1)**	**3.2690 ± 0.2572(1)**	0.0065
CPUsmall	ν-SVR	0.6918 ± 0.0371(5)	0.3120 ± 0.0398(5)	4.8450 ± 0.8492(2)	0.7953
	LS-SVR	0.7140 ± 0.0185(3)	0.2882 ± 0.0192(2)	6.0237 ± 1.3026(5)	10.8428
	ε-TSVR	0.7160 ± 0.0285(2)	0.2994 ± 0.0627(4)	5.6246 ± 1.0662(3)	40.5285
	LIBLINEAR	**0.8579 ± 0.0320(1)**	**0.2138 ± 0.0367(1)**	**1.0918 ± 0.3100(1)**	0.0626
	ν-MADR	0.7107 ± 0.0251(4)	0.2928 ± 0.0281(3)	5.8786 ± 0.5713(4)	0.0183
Bike	ν-SVR	0.3054 ± 0.0237(4)	0.7102 ± 0.0418(4)	**2.0073 ± 0.2880(1)**	33.9155
	LS-SVR	0.3116 ± 0.0108(2)	0.6887 ± 0.0109(2)	2.1121 ± 0.2953(3)	20.2850
	ε-TSVR	0.3088 ± 0.0160(3)	0.6916 ± 0.0167(3)	2.1389 ± 0.3706(4)	60.5827
	LIBLINEAR	0.1456 ± 0.0249(5)	1.2910 ± 0.0973(5)	4.5741 ± 0.4089(5)	0.0725
	ν-MADR	**0.3124 ± 0.0101(1)**	**0.6879 ± 0.0112(1)**	2.0562 ± 0.3886(2)	0.0200
Driftdataset	ν-SVR	0.5838 ± 0.0294(4)	0.4223 ± 0.0341(3)	0.6057 ± 0.0099(4)	58.8105
	LS-SVR	0.4904 ± 0.2335(5)	0.6566 ± 0.4629(5)	**0.5328 ± 0.0617(1)**	105.9514
	ε-TSVR	0.6018 ± 0.0916(3)	0.4035 ± 0.0959(2)	0.5932 ± 0.0931(2)	182.7503
	LIBLINEAR	**0.6219 ± 0.0285(1)**	0.4270 ± 0.0174(4)	0.7941 ± 0.0161(5)	1.5930
	ν-MADR	0.6145 ± 0.0946(2)	**0.3980 ± 0.0456(1)**	0.5966 ± 0.0461(3)	0.6395
Cadate	ν-SVR	0.6179 ± 0.0138(3)	0.3845 ± 0.0162(3)	2.3677 ± 0.3229(3)	49.4465
	LS-SVR	N/A(5)	N/A(5)	N/A(5)	N/A
	ε-TSVR	0.6204 ± 0.0061(2)	0.3799 ± 0.0062(2)	2.3034 ± 0.1976(2)	304.7820
	LIBLINEAR	0.4728 ± 0.0132(4)	0.8760 ± 0.0356(4)	5.0565 ± 1.6076(4)	0.1007
	ν-MADR	**0.6207 ± 0.0158(1)**	**0.3795 ± 0.0169(1)**	**2.1349 ± 0.3066(1)**	0.0356
average rank	ν-SVR	3.8333	3.6667	2.6667	-
	LS-SVR	3.5000	3.3333	3.5000	-
	ε-TSVR	2.5000	2.6667	3.0000	-
	LIBLINEAR	3.5000	4.0000	3.8333	-
	ν-MADR	**1.6667**	**1.3333**	**2.0000**	-



(a) CPU time on large datasets with linear kernel

(b) A closer look at the CPU time in the range of [0,0.7] sec on large datasets

FIGURE 6. The CPU time on large datasets with linear kernel.

TABLE 10. The optimal parameters on large-scale datasets with linear kernel.

Dataset	v-SVR	LS-SVR	ε-TSVR		LIBLINEAR	v-MADR		
	C	C	$c_1 = c_2$	$c_3 = c_4$	C	C	λ_1	λ_2
ConcreteCS	2^{7}	2^{1}	2^{-3}	2^{1}	2^{-9}	2^{9}	2^{-2}	2^{-5}
Abalone	2^{9}	2^{9}	2^{-3}	2^{-1}	2^{-5}	2^{9}	2^{-8}	2^{3}
CPUsmall	2^{-8}	2^{-6}	2^{-9}	2^{7}	2^{-4}	2^{9}	2^{-8}	2^{5}
Bike	2^{5}	2^{-5}	2^{-9}	2^{4}	2^{-9}	2^{9}	2^{4}	2^{3}
Driftdataset	2^{-9}	2^{-6}	2^{-7}	2^{7}	2^{8}	2^{3}	2^{-5}	2^{-9}
Cadate	2^{3}	N/A	2^{-7}	2^{5}	2^{1}	2^{9}	2^{-1}	2^{-7}

For further evaluation, we investigate the absolute regression deviation mean and variance of our v-MADR with RBF kernel, v-SVR, LS-SVR and ε-TSVR on medium-scale datasets as shown in Figure 4. From Figure 4, our v-MADR has the smallest absolute regression deviation mean and variance on most datasets. In addition, v-MADR also has the most compact mean and variance distribution, which demonstrates its robustness. From the above results, it is obvious that our v-MADR outperforms other three methods.

The change of parameter values may have a great effect on the results of regression analysis. For our RBF kernel v-MADR, there are mainly three trade-off parameters, i.e., λ_1, λ_2, C and one kernel parameter σ. Figure 5(a) and Figure 5(b) shows the influence of λ_1 on NMSE and CPU time by varying it from 2^{-9} to 2^{9} while fixing λ_2, C and σ as the optimal ones by cross validation. Figures 5(c)∼5(h) show the influence of λ_2, C and σ on NMSE and CPU time, respectively. As one can see from Figure 5(a), Figure 5(c) and Figure 5(e), the NMSE values on medium-scale datasets do not change significantly when the values of the three parameters λ_1, λ_2, and C are changed. Figure 5(g) shows that σ has more obvious influence on NMSE. On most datasets, as σ becomes larger, NMSE will become smaller and smaller until it converges at a fixed value. Figure 5(b), Figure 5(d), Figure 5(f) and Figure 5(h) show the influence of parameters λ_1, λ_2, C and σ on CPU time. Experimental results indicate that the performance of v-MADR is not sensitive to parameter changes, which further demonstrates the robustness of v-MADR.

B. LARGE-SCALE DATASETS

Table 9 lists the experimental results on six large-scale datasets with linear kernel. We have additionally added a comparison of linear ε-SVR, which was solved by LIBLINEAR [50] that can handle large-scale datasets. In this experiment, because the datasets are too large, for each dataset, 2/3 of the dataset is randomly selected as the training set for feature selection, and the rest 1/3 of the dataset is used as the test set for evaluation. From the average rank at the bottom of Table 9, the overall performance of v-MADR is better than other compared methods or is highly competitive. The optimal parameters are listed in Table 10. Figure 6 shows the comparisons of CPU time. From Figure 6, linear kernel v-MADR is the fastest learning method. In particular, the CPU time of linear kernel v-MADR is far superior to v-SVR, LS-SVR and ε-TSVR.

VI. CONCLUSION

In this research, we introduce statistical information into v-SVR and propose a novel SVR method called v-MADR. v-MADR improves the performance of SVR and overcomes the limitations of existing SVR algorithms by minimizing both the absolute regression deviation mean and the absolute regression deviation variance, which takes into account both the positive and negative values of the regression deviation of sample points. v-MADR proposes a dual coordinate descent (DCD) algorithm for small sample problems, and we also propose an averaged stochastic gradient descent

(ASGD) algorithm for large-scale problems, which greatly reduces the computational complexity and thus improves the algorithm speed. We provide a theoretical analysis on the boundary of the expectation of error for v-MADR. Experimental results have shown that v-MADR outperforms several regression methods and demonstrates great application potential. Our v-MADR Matlab codes can be accessed from: https://github.com/AsunaYY/v-MADR.

In the near future, we will further investigate the potential of v-MADR for big data problems, e.g., predictive analysis for bioinformatics and systems biology problems, and problems in finance. We envision a great application potential in these problems.

REFERENCES

[1] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.
[2] A. J. Smola and B. Scholkopf, *Learning With Kernels*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
[3] J. D. Brown, M. F. Summers, and B. A. Johnson, "Prediction of hydrogen and carbon chemical shifts from RNA using database mining and support vector regression," *J. Biomol. NMR*, vol. 63, no. 1, pp. 39–52, Sep. 2015.
[4] P. K. Rajaraman, T. A. Manteuffel, M. Belohlavek, E. Mcmahon, and J. J. Heys, "Echocardiographic particle imaging velocimetry data assimilation with least square finite element methods," *Comput. Math. Appl.*, vol. 68, no. 11, pp. 1569–1580, Dec. 2014.
[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Oct. 2014.
[6] Y. Ke, B. Fu, and W. Zhang, "Semi-varying coefficient multinomial logistic regression for disease progression risk prediction," *Statist. Med.*, vol. 35, no. 26, pp. 4764–4778, Nov. 2016.
[7] L. H. Dicker, "Ridge regression and asymptotic minimax estimation over spheres of growing dimension," *Bernoulli*, vol. 22, no. 1, pp. 1–37, Feb. 2016.
[8] R. M. Balabin and E. I. Lomakina, "Support vector machine regression (SVR/LS-SVM)—An alternative to neural networks (ANN) for analytical chemistry? Comparison of nonlinear methods on near infrared (NIR) spectroscopy data," *Analyst*, vol. 136, no. 8, pp. 1703–1712, 2011.
[9] I. Khosravi, Y. Jouybari-Moghaddam, and M. R. Sarajian, "The comparison of NN, SVR, LSSVR and ANFIS at modeling meteorological and remotely sensed drought indices over the eastern district of isfahan, iran," *Natural Hazards*, vol. 87, no. 3, pp. 1507–1522, Jul. 2017.
[10] N. Suyaroj, N. Theera-Umpon, and S. Auephanwiriyakul, "A comparison of NN-based and SVR-based power prediction for mobile DS/CDMA systems," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst.*, Feb. 2009, pp. 1–4.
[11] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.
[12] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Comput.*, vol. 12, no. 5, pp. 1207–1245, May 2000.
[13] J. A. K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle, "Least squares support vector machine classifiers: A large scale algorithm," in *Proc. Eur. Conf. Circuit Theory Design*, 1999, p. 99.
[14] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.
[15] X. Peng, "TSVR: An efficient twin support vector machine for regression," *Neural Netw.*, vol. 23, no. 3, pp. 365–372, Apr. 2010.
[16] Y. H. Shao, C. H. Zhang, Z. M. Yang, L. Jing, and N. Y. Deng, "An μ-twin support vector machine for regression," *Neural Comput. Appl.*, vol. 23, no. 1, pp. 175–185, 2013.
[17] Z.-M. Yang, X.-Y. Hua, Y.-H. Shao, and Y.-F. Ye, "A novel parametric-insensitive nonparallel support vector machine for regression," *Neurocomputing*, vol. 171, pp. 649–663, Jan. 2016.
[18] S. Balasundaram and G. Benipal, "On a new approach for lagrangian support vector regression," *Neural Comput. Appl.*, vol. 29, no. 9, pp. 533–551, May 2018.
[19] S. Balasundaram and D. Gupta, "On implicit lagrangian twin support vector regression by Newton method," *Int. J. Comput. Intell. Syst.*, vol. 7, no. 1, pp. 50–64, Jan. 2014.
[20] T. Zhang and Z.-H. Zhou, "Large margin distribution machine," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 313–322.
[21] Z.-H. Zhou, "Large margin distribution learning," in *Proc. 6th Int. Workshop Artif. Neural Netw. Pattern Recognit.*, vol. 8774, 2014, pp. 1–11.
[22] T. Zhang and Z.-H. Zhou, "Optimal margin distribution machine," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1143–1156, Jun. 2020.
[23] W. Gao and Z.-H. Zhou, "On the doubt about margin explanation of boosting," *Artif. Intell.*, vol. 203, pp. 1–18, Oct. 2013.
[24] M.-Z. Liu, Y.-H. Shao, Z. Wang, C.-N. Li, and W.-J. Chen, "Minimum deviation distribution machine for large scale regression," *Knowl.-Based Syst.*, vol. 146, pp. 167–180, Apr. 2018.
[25] R. Rastogi, P. Anand, and S. Chandra, "Large-margin distribution machine-based regression," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3633–3648, Apr. 2020.
[26] P. Anand, R. Rastogi, and S. Chandra, "Generalized ϵ-loss function-based regression," in *Proc. Mach. Intell. Signal Anal.*, 2019, pp. 395–409.
[27] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
[28] W. Wang, "An incremental learning strategy for support vector regression," *Neural Process. Lett.*, vol. 21, no. 3, pp. 175–188, Jun. 2005.
[29] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 7, pp. 1403–1416, Jul. 2015.
[30] B. Gu and V. S. Sheng, "A robust regularization path algorithm for μ-support vector classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 5, pp. 1241–1248, May 2017.
[31] D. J. Crisp and C. J. C. Burges, "A geometric interpretation of v-SVM classifiers," in *Proc. 13th Annu. Neural Inf. Process. Syst. Conf.*, pp. 244–250, 2000.
[32] J. S. Marron, "Distance-weighted discrimination," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 7, no. 2, pp. 109–114, Mar. 2015.
[33] B. Schölkopf, "Learning with kernels: Support vector machines, regularization, optimization, and beyond," *Amer. Stat. Assoc.*, vol. 98, no. 462, p. 489, 2003.
[34] A. F. Izmailov and M. V. Solodov, "Karush-kuhn-tucker systems: Regularity conditions, error bounds and a class of Newton-type methods," *Math. Program.*, vol. 95, no. 3, pp. 631–650, Mar. 2003.
[35] D. Anguita, A. Ghio, S. Pischiutta, and S. Ridella, "A support vector machine with integer parameters," *Neurocomputing*, vol. 72, nos. 1–3, pp. 480–489, Dec. 2008.
[36] L. Oneto, A. Ghio, S. Ridella, and D. Anguita, "Learning resource-aware classifiers for mobile devices: From regularization to energy efficiency," *Neurocomputing*, vol. 169, pp. 225–235, Dec. 2015.
[37] L. Oneto, J. L. R. Ortiz, and D. Anguita, "Constraint-aware data analysis on mobile devices: An application to human activity recognition on smartphones," in *Adaptive Mobile Computing*. Salt Lake City, UT, USA: Academic, Oct. 2017, pp. 127–146, doi: 10.1016/B978-0-12-804603-6.00007-3.
[38] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 408–415.
[39] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proc. IEEE*, vol. 100, no. 9, pp. 2584–2603, Sep. 2012.
[40] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM J. Control Optim.*, vol. 30, no. 4, pp. 838–855, Jul. 1992.
[41] H. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. New York, NY, USA: Springer, 2003.
[42] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. 19th Int. Conf. Comput. Statist.*, Paris, France, 2010, pp. 177–186.
[43] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, pp. 919–926.
[44] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Math. Program.*, vol. 127, no. 1, pp. 3–30, Mar. 2011.
[45] B. Antoine, B. Leon, and G. Patrick, "SGD-QN: Careful quasi-Newton stochastic gradient descent," *J. Mach. Learn. Res.*, vol. 10, pp. 1737–1754, Dec. 2009.

[46] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 71–79.
[47] W. Xu, "Towards optimal one pass large scale learning with averaged stochastic gradient descent," 2011, *arXiv:1107.2490*. [Online]. Available: https://arxiv.org/abs/1107.2490
[48] A. Luntz, "On estimation of characters obtained in statistical procedure of recognition," *Technicheskaya Kibernetica*, vol. 3, pp. 6–12, 1969. [Online]. Available: https://scholar.google.com/scholar_lookup?title=On%20estimation%20of%20characters%20obtained%20in%20statistical%20procedure%20of%20recognition&publication_year=1969&author=A.%20Luntz
[49] C.-C. Chang and C.-J. Lin. (2018). *LIBSVM—A Library for Support Vector Machines*. [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/libsvm
[50] Machine Learning Group at National Taiwan University. (2019). *LIBLINEAR—A Library for Large Linear Classification*. [Online]. Available: https://www.csie.ntu. edu.tw/~cjlin/liblinear
[51] K. Pelckmans. (2002). *LSSVMlab: A MATLAB/C Toolbox for Least Square Support Vector Machines*. [Online]. Available: http://www.esat.kuleuven.ac.be/sista/lssvmlab
[52] Y.-H. Shao, C.-H. Zhang, X.-B. Wang, and N.-Y. Deng, "Improvements on twin support vector machines," *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 962–968, Jun. 2011.
[53] O. L. Mangasarian and D. R. Musicant, "Successive overrelaxation for support vector machines," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1032–1037, Oct. 1999.
[54] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

YAN WANG received the Ph.D. degree from the College of Computer Science and Technology, Jilin University, China, in 2007. He was doing research on a bioinformatics collaborative project as a Visiting Scholar and a Postdoctoral Researcher at the University of Georgia, USA, from 2007 to 2011. From 2012 to 2013, he held a postdoctoral position at the University of Trento, Italy. He is currently a Professor and the Ph.D. Supervisor with the College of Computer Science and Technology, Jilin University. His research interests focus on computational intelligence, bioinformatics, and e-business, such as machine learning and deep learning, big data intelligent computing, gene expression analysis, cancer biomarker prediction, and overlapping community detection. He has over 70 research articles published, including more than 30 indexed by SCI, such as *Nucleic Acids Res*, *Briefings in Bioinformatics*, *Computational and Structural Biotechnology Journal*, *Frontiers in Genetics*, *Scientific Reports*, *Applied Soft Computing*, and *Bioinformatics*. He has also presided over or taken part in several projects funded by the National Natural Science Foundation and the 863 Project in China.

YAO WANG received the master's degree from the College of Computer Science and Technology, Jilin University, China, in 2019, where he is currently pursuing the Ph.D. degree. He is also a Student with the Jilin Provincial Key Laboratory of Big Data Intelligent Computing. His research interests are mainly in the field of machine learning, deep learning, and medical image processing.

YINGYING SONG received the bachelor's degree from the College of Computer Science and Technology, Jilin University, China, in 2017, where she is currently pursuing the master's degree. She has published two articles and won two school scholarships. Her research interests are mainly in the field of machine learning, regression analysis, feature selection, and deep learning.

XUPING XIE received the bachelor's degree from the College of Information Science and Technology, Northeast Normal University, in 2019. She is currently pursuing the master's degree with the College of Computer Science and Technology, Jilin University, China. She is also a Student with the Jilin Provincial Key Laboratory of Big Data Intelligent Computing. Her research interests are mainly in the field of machine learning and deep learning.

LAN HUANG received the B.S., M.S., and Ph.D. degrees in computer science and technology from Jilin University, Changchun, in 1994, 1999, and 2003, respectively. She is currently a Professor, a Supervisor of Ph.D. Students with Jilin University. She is the Director of the Key Laboratory of Big Data Intelligent Computing, Jilin. She is mainly engaged in intelligent computing, data mining theory and application research, and high-performance computing. She is an Outstanding Member of CCF. She won the middle-aged and young experts with outstanding contributions in Jilin. As a PI and Co-PI, she has been undertaking or accomplished more than ten teaching and scientific research projects, granted by the National 863 Hi-Tech Research and Development Program, the National Science Foundation China, Provincial/Ministerial Foundations, and other sources. The projects that she participated as a main investigator were awarded the first prizes for the Jilin Province Scientific and Technological Progress Award (First Prize Winner, in 2017, and the Second Prize Winner, in 2019). In recent five years, she has published more than 80 academic articles and won 1 Second Prize for teaching achievements of Jilin, in 2018, and published 4 textbooks as the Chief Editor.

WEI PANG received the Ph.D. degree in computing science from the University of Aberdeen, U.K., in 2009.

He is currently an Associate Professor with Heriot-Watt University, Edinburgh, U.K. He is also an Honorary Senior Lecturer with the University of Aberdeen. He has authored over 100 articles, including more than 40 journal articles. His research interests include bio-inspired computing, data mining, machine learning, and qualitative reasoning. He was a recipient of the Best Paper Award in the 19th Annual U.K. Workshop on Computational Intelligence (UKCI 2019) and the Best Paper Runner Up Award in the 12th International Conference on Advanced Data Mining and Applications (ADMA 2016).

GEORGE M. COGHILL received the Ph.D. degree in fuzzy qualitative reasoning from the Intelligent Systems Laboratory, Heriot-Watt University. He was promoted to a Reader of Aberdeen in computing science, in 2009, and the Chair, in 2011. At Aberdeen, he has undertaken a number of roles. He was the Head of Computing Science, from 2010 to 2014, the Director of Research, and the Deputy Head of the School of Natural and Computing Sciences, from 2015 to 2018. His research interests are mainly in the field of model-based systems and qualitative reasoning. He has served on a several committees in the institution, including the Academic Standards Committee (Postgraduate), the Quality Assurance Committee, and the Research Policy Committee.